[redacted]

Il6r
o.944
op.2 UIUCDCS-R-78-944

Math

UILU-ENG 78 1737

TRAINABLE CHARACTER RECOGNITION INTERFACE COMPUTER (INCOM)

by

Mohamed Taher Abdalla El-Sonni

© 1978

October 1978

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-78-944

TRAINABLE CHARACTER RECOGNITION INTERFACE COMPUTER (INCOM)

by

Mohamed Taher Abdalla El-Sonni

October 1978

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

Supported in part by the Department of Computer Science, and submitted in partial fulfillment of the requirements of the Graduate College for the degree of Doctor of Philosophy.

TRAINABLE CHARACTER RECOGNITION INTERFACE COMPUTER (INCOM)

Mohamed Taher Abdalla El-Sonni, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1978

A trainable, real-time character recognition device has been designed and built. The visual feature extraction method is chosen as the most favorable computational approach around which the feature extractor is designed. A new method of local feature extraction is presented which uses a minimal size window and simple raster scanning of the character image. Merging rules are devised to reduce the number of these features by merging two features at a time. A method of dynamic segmentation of the character representations is introduced, in which the character is described as a collection of horizontal or vertical segments connected by links. A multiple description technique of the character segments make it possible to reduce the number of training characters and to use a simple data structure for the classification dictionary as well as a simple decision criterion for recognition. Experiments show the power of the described techniques.

ACKNOWLEDGEMENT

I wish to express my gratitude to my advisor, Professor Michael Faiman, for suggesting the thesis topic and for his continuous guidance and friendship. I am also grateful to Professor Wolfgang Poppelbaum for the opportunity to work in the Information Engineering Laboratory and for his friendship. I thank Professors Sylvian Ray and William Kubitz for encouragement and friendship.

A special word of thanks goes to Frank Serio for his expertise in fabricating the printed circuit boards and the panel, Stan Zundo for his personal care and skill in drafting the figures, Cinda Robbins for her assistance and for typing the final draft, Clyde Helm for cooperation and to June Wingler for her excellent job in typing the thesis.

I am indebted to my friend Professor Ahmed Sameh for his most appreciated advice and encouragement. I would like also to thank my friends in the Information Engineering Laboratory: Al Irwin, Gary Gostin, Dan Pitt, Joe Luhukay, Mike Robinson, Izumi Suwa and Randy Moss for their friendship and for many valued discussions.

Finally, I wish to express my special thanks to my wife Magda for typing the early drafts of the thesis and especially for most needed encouragement and support, and to my son Taher for encouragement in his special way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Design Goals

INCOM (INterface COMputer) is a real time, trainable hand-drawn symbol recognition device. It is intended as an interface between a graphics input tablet and a computer system in an interactive graphics environment. INCOM can be trained on-line to recognize hand-drawn symbols such as alphanumerics, flowchart symbols and symbols of similar complexity. In addition, the average number of training samples is reasonably small: about 10 samples/symbol for a recognition rate of 80% or better. The system response, in training or recognition is almost instantanious (about 50 msec).

The device, as it stands, can be regarded as a character recognition machine, and a character recognition technique has been chosen and implemented in hardware. Such a device will certainly save the computer system the burden of recognizing the input character besides the obvious speed up of the recognition task. There is also the advantage of having a character recognition machine of reasonable size which can be used for man-machine interaction purposes, personal use, e.g. reading for the blind, or as the recognition part of an optical character recognition device. It is interesting to note that there is no reported device which has been designed and implemented specifically to be trained on-line to recognize hand-drawn symbols or characters. Actually most of the known approaches in character recognition have been implemented on large computers [1]. Several techniques have employed a front-end processor, a special purpose computer, or a minicomputer, to pre-process the input data and rely on a bigger computer to store the recognition dictionary and to handle the training [5].

1.2 Relevant Literature

Character Recognition has been the subject of a large number of papers for almost two decades [1,2,3,4]. Actually almost every approach to pattern recognition has been tested on samples of characters to show some degree of applicability. Character recognition methods can be classified in several ways. For the purpose of this thesis these will be discussed according to the following three major requirements: accessibility to the user; expandability of the character set; and the type of computation used in the recognition process.

In terms of user accessibility, both off-line and on-line (real-time) approaches have been considered [1]. In the former there is no interaction between the user and the device; optical character recognition falls under this heading. In the on-line approach, the user interacts with the device by inputting, modifying and even training the system to recognize his own symbols. We are interested in this latter approach.

According to the expandability of the character set the device can be characterized as of the specialized type or the trainable type. If specialized, the device is tuned to recognize a limited set of characters which the user may not change. INCOM, on the other hand, is trainable to recognize characters specified by the user. Trainable systems are more flexible, but may exhibit some degradation of recognition if used by others without retraining.

A character recognition device can be viewed as consisting of three main stages: preprocessor, feature extractor and classifier (Figure 1.1). The preprocessor operates on the raw input data preparing it for further processing. A set of measurements are then performed on the preprocessed pattern image by the feature extractor, which produces a compacted form of these measurements. We call this form a feature vector. To train the device, the resultant feature vector is tagged with the input character class and presented to the classifier. The classifier then stores this information into its data structure. In the testing stage (recognition) the feature vector of the unknown character is presented to the classifier. The classifier uses the feature vector to retrieve relevant information from the classifier data structure. According to some decision criterion, the retrieved information is used to find the unknown character class.

To obtain good performance from a character recognition device the set of measurements performed by the feature extractor must be relevant to the classes of characters to be recognized. This makes the feature extractor stage the most important and critical stage of any character recognition device. Therefore, the computational approaches will be categorized here according to the types of measurements used in order to construct representations of input character patterns. These may take any of the following forms: template matching, pattern sampling, spatial transform, geometrical moments

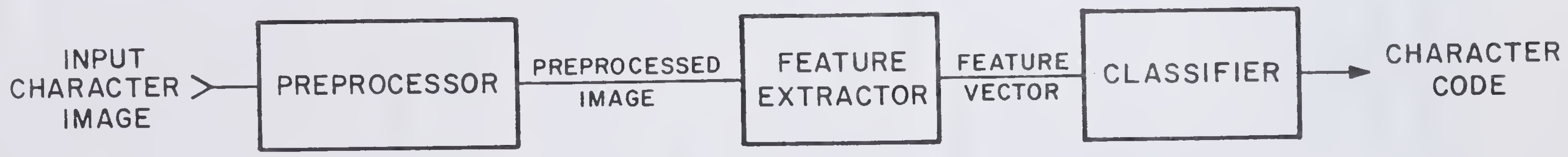


Figure 1.1 General Block Diagram of a Character Recognition Device

and visual feature extraction.

Template matching [1] is the simplest of all the computational methods. Measurements are taken directly from the character image and, therefore, the resultant features are the points constructing the image. In the learning phase the character prototypes are stored in the classifier dictionary. In the recognition phase the input character image is matched against the character prototypes using a decision criterion. The character prototype closest to the input character according to this criterion is considered the recognized character. This technique has been applied successfully to the problem of fixed-font optical character recognition. It is not suitable for hand-printed character recognition because of the large number of prototypes required to accommodate the many variations in characters.

Because of the inherent redundancy in human generated symbols we can expect that only a fraction of the points constructing the character images actually contribute to the recognition process. It should be sufficient to sample the input character pattern at some known places in the image plane and use these samples, e.g. groups of points, as measurements. The measurements do not necessarily represent the character image faithfully but can nevertheless be useful in the recognition task. These measurements can be the number of points in some chosen fixed areas of input image, points of intersections between image lines and lines of known directions [6], the presence or absence of certain points in the image [7], etc. This method is called pattern sampling. It has obvious advantages over the template matching approach by allowing more variations in the input character. But both of them have the disadvantage of requiring large numbers of training prototypes, especially for the kind of patterns we are dealing with: skeletonized patterns or character images from an input tablet. However, pattern sampling has been tested successfully on optical character recognition systems for letter sorting [8].

A third technique is to use a set of orthogonal functions, e.g., Fourier, Walsh/Hadamard, Haar, etc. [9,10] to spatially transform the input image, and use the resultant transform coefficients as measurements. Because of the large number of the resultant coefficients, n^2, where n is the dimension of the input matrix, some of them are chosen under program control, hoping that they will be sufficient to recognize the unknown input patterns. The drawback here lies in the choice of these coefficients, which require a large sampling of input patterns, as well as a powerful computer with a sufficiently large memory [10]. It works well if the number of character classes is limited and the variations in the unknown input characters are not severe. For example, if the unknown character is skewed or shifted in one direction the transform coefficients will change values appreciably. This will result in either misclassification or rejection.

A fourth approach is that of weighting the different points of the input character image with known weight functions and summing the result, an instance of which is the method of geometrical moments [11]. Although these measurements can be made translation-invariant, this technique possesses the major disadvantage of the spatial transform method: large amount of training data and insensitivity to local variations of character classes.

The fifth scheme is that of visual feature extraction, in which a character is considered as a collection of interrelated meaningful features. For example, a character can be described as a collection of line strokes [5,12], or as a collection of line endings (spurs), line-crossing, etc. [13]. A "feature" represents a set of characteristics of the image plane that is invariant to several character classes and is also invariant to translation. Therefore, describing a character using these kinds of features will inherently describe variations of this character, thus enhancing recognition. Clearly, the choice of appropriate features, as well as methods of extracting and interrelating them, are crucial to the efficiency of any character recognition device which uses this scheme.

1.3 Comparison of Computational Methods

Before adopting a computational technique for implementation, three general criteria are considered: (i) amount of storage per character set, (ii) amount of processing per sample, and (iii) number of training samples per character class. Table 1.1 gives the ranking (1-5) of the five computational approaches. Rank 1 indicates the most favorable and rank 5 is the least favorable, according to size, complexity, cost, speed, etc.

The first criterion, amount of storage per character set, is an indication of the complexity of the system as a whole and its response. It includes the storage required for programs, intermediate results and classifier dictionaries, as well as the amount of processing

Table 1.1 Ranking of Computational Methods According to Comparison Criteria.

Approaches	Amount of Storage/ Character Set	Amount of Processing/ Sample	Training Samples/ Character Class
Template Matching	5	1	5
Pattern Sampling	4	2	4
Spatial Transforms	3	5	3
Geometrical Moments	2	4	2
Feature Extraction	1	3	1

required for the relevant measurements. The feature extraction method is ranked first because the measurements need less storage and processing than the rest. In addition, the size of the classification dictionaries is usually much less than that of the other schemes. The large sizes of the classification dictionaries and the amount of processing required for the associated measurements make template matching and pattern sampling the least favorable. Geometrical moments have some advantages over the spatial transform approach because the resulting moments are more locally sensitive and can be made translation invariant and, hence, better recognition characteristics can be expected.

The second criterion, amount of processing per sample, indicates the amount of computation required for performing the measurements on the input character. The top two methods in Table 1.1 are simpler and require less computation than the rest. Feature extraction, however, is superior to spatial transforms and geometrical moments. For the case of line-like patterns it is computationally simpler and requires less time to extract features than generating transform coefficients or geometrical moments.

The third criterion, number of training samples per character class, is a direct requirement from the design goals: the smaller the number of training samples the better the approach. The feature extraction approach excels in this requirement as mentioned in section 1.2.

From the table it is obvious that feature extraction has the highest overall ranking and is therefore the method of choice for INCOM.

1.4 Design Strategy

After adopting the feature extraction approach, it is necessary to determine which features and interrelationships will be used. To achieve such a goal, first, human factors must be considered, since they are involved in generating and interpreting characters. Secondly, in detecting the chosen features, it would be advantageous to exploit the input form, i.e., binary pattern, of the character image.

a. The Human Factors

(i) The lines drawn are affected by the complex motor activity of the hand as well as its inertia. Because of this, lines people draw are not necessarily straight, even if they are meant to be so, and lines meant to meet at a vertex may cross each other or may be slightly separated. A good technique has to tolerate such deviations and eliminate some of them before further processing. It has also been observed that obvious changes in the line direction best describe line-like scenes [14]. Line-curvature, line-endings and line intersections have been chosen as the local features of characters for implementing INCOM. Before detecting these features gaps are filled between two points considered close enough to be connected.

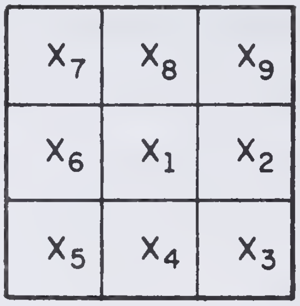
(ii) It has been observed in experimental psychology, using human subjects, that segmenting the character horizontally gives a good description [14]. Actually this segmentation has already been used in a practical system to recognize hand-written numerical characters for automatic letter sorting [15]. Another approach is to segment the character image plane into horizontal, vertical and diagonal strips of known positions and detect the presence of simple features, like short lines and long lines, within these strips. This approach has been applied successfully to recognize alphanumeric characters [16].

In INCOM horizontal as well as vertical segmentation are done dynamically on the character representations. Segmentation here is made dependent on the distribution of local features within the character itself, instead of the fixed segmentation of the image plane of the other approaches. In addition, the resulting segments are themselves segmentable for the purpose of establishing desirable topological relationships with other parts of the character.

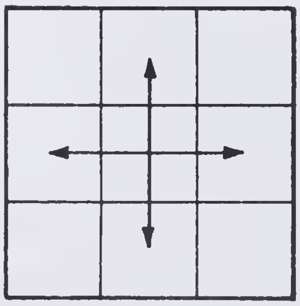
(iii) Horizontal and vertical orientations are preferred to others in describing the spatial relationship between objects [17]. In INCOM relationships like "above," "below," "left of" and "right of" are used to describe spatial interrelationships between parts of a hand drawn character. However, these descriptions are not explicitly coded but they are imbedded in the feature vector describing the character.

b. The Input Form

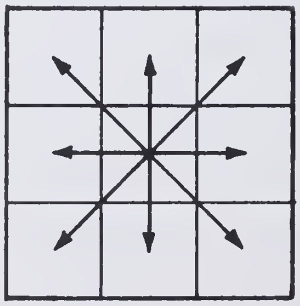
Since the input from the tablet is a series of points to be stored in a matrix storage, an effort has been made to exploit the binary representation of line-like drawings. The input points are stored as 1-cells, while 0-cells represent the unwritten part of the image matrix. Let $x_2, x_3, \ldots, x_9$ denote the cells (1's or 0's) neighboring an arbitrary center cell x_1 in Figure 1.2(a). Two types



a) Window Cells Designation



b) 4-Connectivity



c) 8-Connectivity

Figure 1.2 Window Cells Designation and Types of Connectivities

of connectivity: 4-connectivity and 8-connectivity, are defined between a neighboring cell and the center cell when both of them are 1-cells as shown in Figure 1.2(b) and (c). Two points in a binary image are connected when there is a series of pairs of 4-connected or 8-connected points between them. To simplify the feature extraction operation the binary image of the input character is thinned by changing superfluous 1-cells to 0-cells on the edges of lines, while preserving connectivity and tips of lines. After thinning (cleaning) the binary image, it may be observed that 3×3 cells is the smallest window through which a change of line direction can be detected.

Using the above properties three sets of local patterns of 3×3 binary matrices have been constructed. These are the gap-filling, image cleaning and nodal extraction templates for connecting close parts of the binary image, eliminating redundant points and extracting local topological features (e.g., tips of lines, change of direction, etc.).

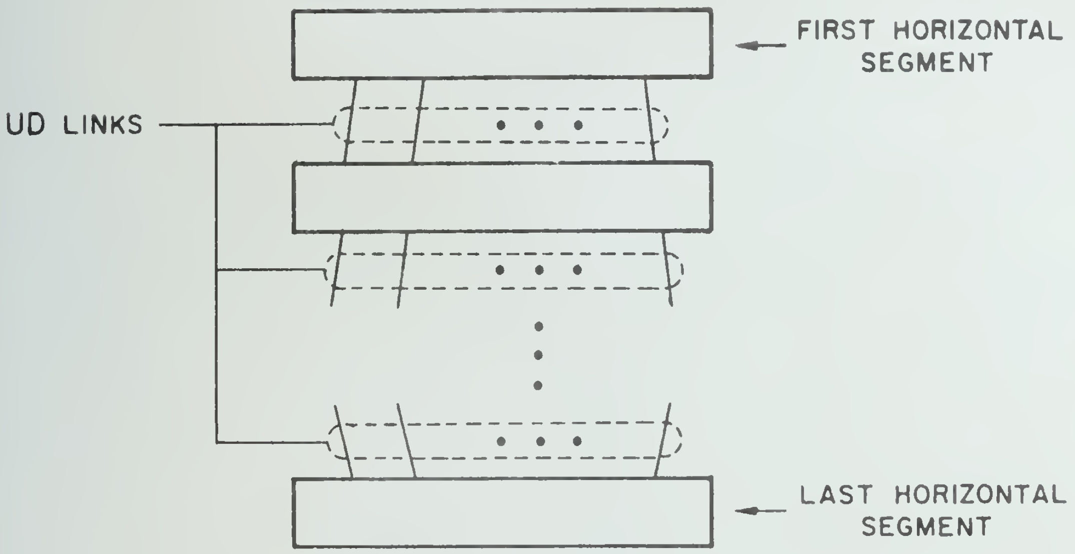
2. SYSTEM DESIGN

2.1 Introduction

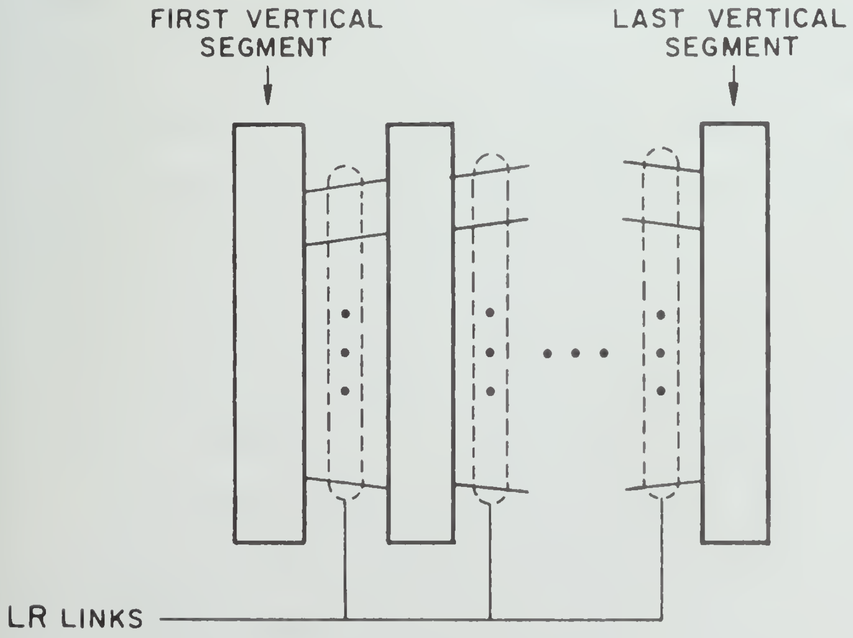
In INCOM a character is viewed as a set of local features connected by a set of links. In other words, it may be regarded as a linear graph with local features as vertices, or nodes, and links as edges. Instead of using lists of node coordinate information and explicit searching for connections between them, a simple method is described for specifying the character (graph) by means of a set of one-dimensional list subvectors. This method is segmentation, in which the character is described as a collection of either horizontal or vertical segments, which are connected by links (Figure 2.1). Segments may consist of several disconnected subsegments each of which may contain one or more nodes.

For the current implementation minimum descriptions of the constitutent parts of the character--segments, links, and nodes--have been considered. The existence of segments and subsegments, the number of links and their distribution within subsegments and the number of nodes in each subsegment constitute the feature vector.

It should be noted that all the operations involved in preprocessing and feature extraction as well as constructing the feature vector are done by simple raster scanning of the WORKING STORAGE memories without resorting to contour following or back tracking. This makes the control simpler and the implementation more elegant than the other approaches.



a) Horizontal Segmentation



b) Vertical Segmentation

Figure 2.1 Character Segmentation

In addition, the multiple-description approach makes it easier to devise a simple data structure for learning and a simple best fit criterion for decision (recognition).

2.2 General Description

INCOM consists functionally of three main processors: the PREPROCESSOR, the FEATURE EXTRACTOR and the CLASSIFIER (Figure 2.2). The PREPROCESSOR accepts a series of points (x,y coordinates) from a tablet digitizer, or its equivalent, and stores them in the image memory (IMAGEM). Upon finishing the drawing of the input character the PREPROCESSOR begins the operation of gap-filling followed by image-cleaning, thus preparing for feature extraction operations. The FEATURE EXTRACTOR operates on IMAGEM by extracting local features and storing them in the node memory (NODEM). Furthermore, according to the current scanning mode the vertically or horizontally inclined links are extracted from IMAGEM and stored in LINKM. The WORKING STORAGE memories (IMAGEM, NODEM and LINKM), which now represent the input character, are scanned either horizontally or vertically. While scanning, they are partitioned into segments. Each segment is represented by the links which connect it to the neighboring segments and the number of local features in each segment. Each segment is then coded as elements of three feature subvectors, ILINKV, OLINKV and TABV. The locations of these elements are indicative of the order of their corresponding segments. The feature vector which consists of these three subvectors is the input to the CLASSIFIER.

The CLASSIFIER preprocesses the feature vector by combining parts of its constituents into compound features (V-ELEMENTS) according

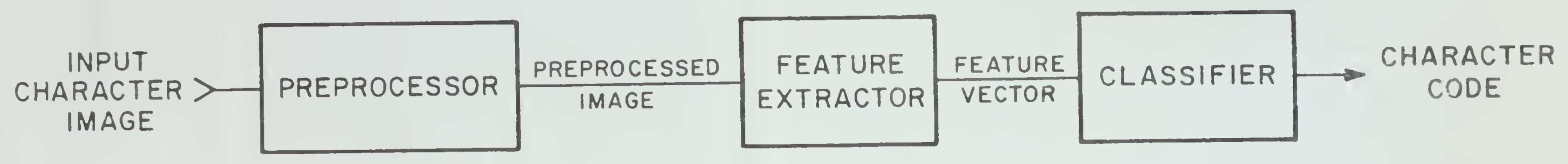


Figure 2.2 INCOM as a Character Recognition Device

to a pre-determined algorithm. In the learning mode, training, each newly constructed V-ELEMENT is stored in conjunction with an updated list of the character classes in which it has appeared. In the recognition mode the classifier memory, which holds the old V-ELEMENTS, is searched for a match with the new V-ELEMENT. When a match is found the associated character classes are read in a special set of registers. The character class which scores the most matchings or posseses a unique V-ELEMENT (one which has not appeared in any character while training) is considered the recognized character.

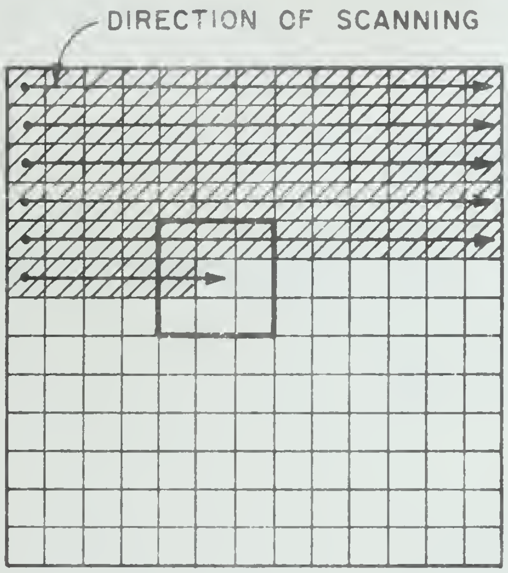
2.3 Scanning-Windowing Schemes

To process the WORKING STORAGE three main scanning-windowing schemes are used. Two are of the raster scan type, in which a fixed size window scans a whole memory plane either row by row or column by column. The third is a local scanning scheme in which a group of cells in a window are scanned in anticlockwise direction. The three schemes are designated as HSCANWmn, VSCANWmn and ASCANNmn schemes for horizontal, vertical and anticlockwise scanning, respectively, using mxn windows.

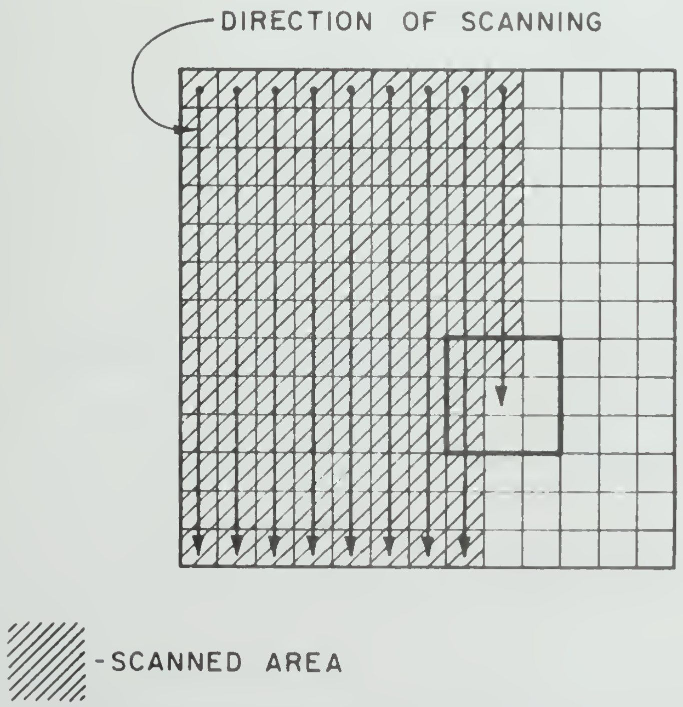
Figure 2.3 shows examples of the first two schemes using a 3x3 window. Figure 2.4 shows the scheme used in merging.

2.4 The PREPROCESSOR

Three operations are performed on IMAGEM: points storing, gap-filling and image-cleaning. In this section we will assume that the binary image is already stored in IMAGEM.



a) HSCANW33



b) VSCANW33

Figure 2.3 Horizontal and Vertical Scanning-Windowing Schemes using 3x3 Window

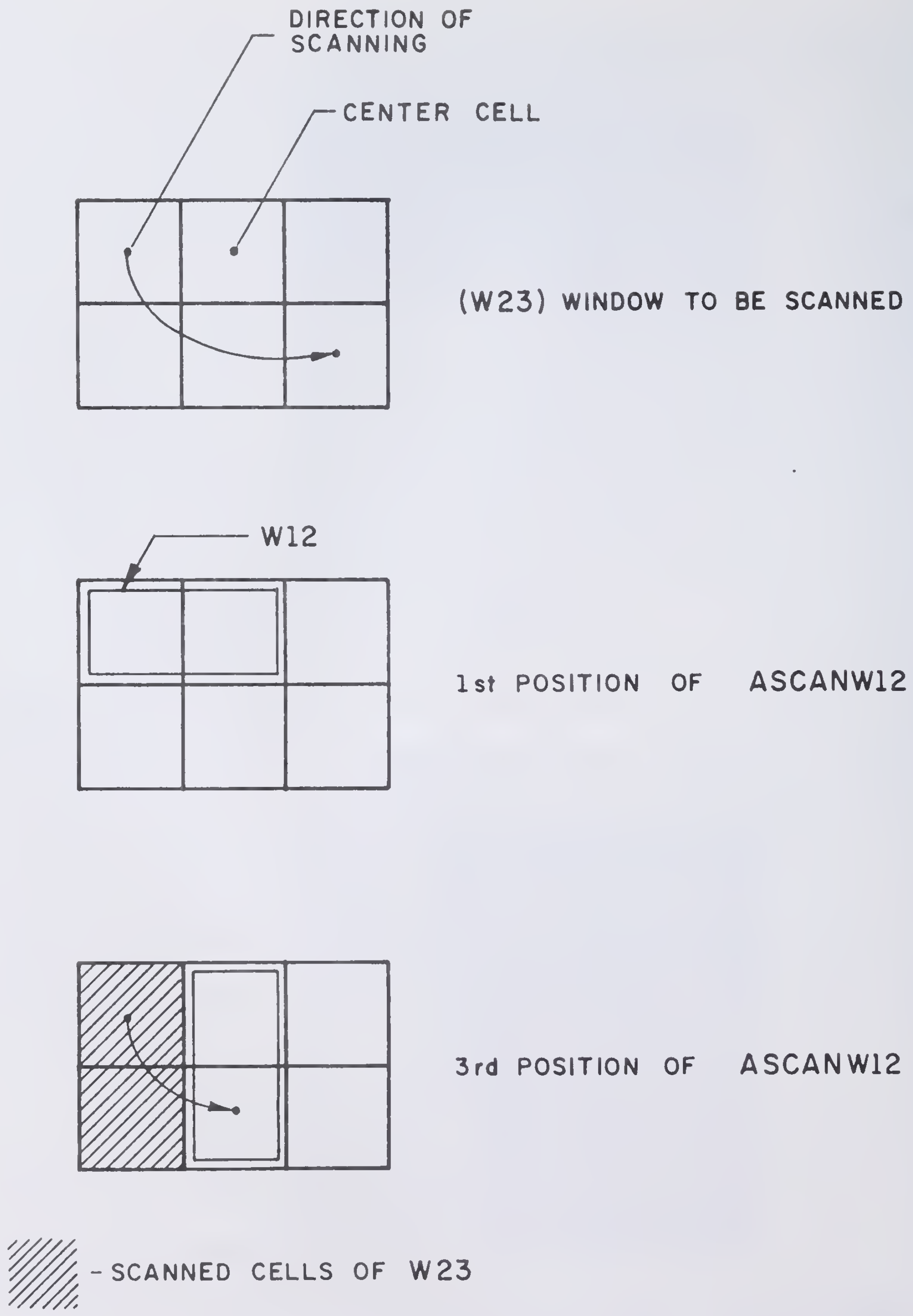


Figure 2.4 Anticlockwise Scanning-Windowing Scheme using 1x2 Window

IMAGEM is a two-dimensional 16x16 cells matrix for storing the binary image of the input character. The image is stored in the inner 14x14 matrix leaving the outer rows and columns empty (0-cells) for ease of manipulation.

We consider 14x14 a reasonable size for the binary pattern under consideration. As a matter of fact a matrix of 14 rows by 9 columns has been found satisfactory for representing character images (17).

Gap-filling is the process by which gaps (0-cells) are filled with points (1-cells) between closely adjacents points, or between a point and a line or tips of lines. A gap may occur as a result of quantizing the input drawing on a discrete grid. It can also occur because the user may consider the gap insignificant (see section 1.4 on Human Factors). One cell is considered a reasonable gap in a 16x16 cells IMAGEM. This assumption makes the gap-filling easy to perform using HSCANW33 (section 2.3). Examples of fill-in-gap patterns (FIG) are shown in Figure 2.5(a).

For gap-filling the center cell of a 3x3 pattern is a 0-cell while the neighboring cells may take any combination of 0-cells and 1-cells; 256 combinations in all. A FIG-Table of 256 1-bit entries has been constructed. Each entry is addressed by the contents of the outer cells of the current window. If the pattern is a FIG pattern the addressed entry contains '1'; otherwise it contains '0'. To fill gaps, IMAGEM is scanned using the HSCANW33 scheme and the center cell of the current window is changed to '1' if the surrounding cells address a FIG pattern in the table. Otherwise, it remains the same

0	1	1
0	0	0
1	0	0

0	1	1
0	0	1
1	0	0

1	0	0
1	0	1
1	0	0

a) Fill-in-gap (FIG) Patterns

0	0	0
1	1	0
0	1	0

0	0	0
0	1	0
1	1	1

0	0	1
0	1	1
0	1	1

b) Clean-image (CLN) Patterns

Figure 2.5 Examples of Preprocessing Patterns

and the next window location is considered. The FIG operation is completed in one IMAGEM scan.

The clean-image process (CLN) follows the fill-in-gap (Figure 2.5(b)), and is designed to get rid of some of the quantization noise by smoothing staircase-like lines and deleting one-cell extensions of intersecting lines. In cleaning, the center cell under consideration is reset to '0'. The same scanning table look-up mechanism used for fill-in gap is also used here. A CLN-Table is constructed: an entry contains '0' if it corresponds to a CLN pattern, otherwise it contains '1'. While scanning IMAGEM the center cell will be changed to '0', cleaned, if the surrounding cells address a CLN pattern, otherwise it remains the same. The clean-image operation also needs one scan only.

After the above two operations, the binary pattern in IMAGEM is suitable for feature extraction.

2.5 The FEATURE EXTRACTOR

The feature extraction technique adopted here is based on the hierarchical description of characters represented by multiple descriptions of the components. A character can be described as a human-generated line-like pattern which can be segmented into smaller patterns (segments) inter-related by connectivity (links) and spatial relations (above, left-of, etc.). Each segment in turn may consist of subsegments. Each segment is described by the distribution of links connecting it to its neighboring segments, by its contents of local features (ends of lines, curved lines, etc.), their distributions in the segment, etc. Each description is encoded and stored separately in a feature subvector.

In INCOM each character is segmented horizontally and vertically. For each segment three descriptions are given: two descriptions using the distribution of the lines linking it to the neighboring segments (ILINKV and OLINKV), and the distribution of local features in each segment (TABV).

Three main operations are performed in the FEATURE EXTRACTOR: local feature extraction, structural feature extraction and feature vector formation. Local feature extraction involves two consecutive operations: one on IMAGEM to extract nodes and store them in NODEM and the other operation (Merging) is applied on NODEM to eliminate the superfluous nodes. Both of these operations are explained in the next two sections.

2.6 Local Feature Extraction

a. Node Extraction

The types of nodes and their line representations as well as examples of their equivalent binary patterns are shown in Figures 2.6 and 2.7. It has been observed that 3x3 cells is the smallest window through which line endings, changes of direction and lines intersection can be detected (section 1.4). The image cleaning operation has made it possible to use such a small size window to detect these nodes. The same scanning-windowing scheme, HSCANW33, in conjunction with addressing a table, NEX-TABLE, is used here to detect the nodes in IMAGEM. The contents of NEX-TABLE are the class codes of the nodes.

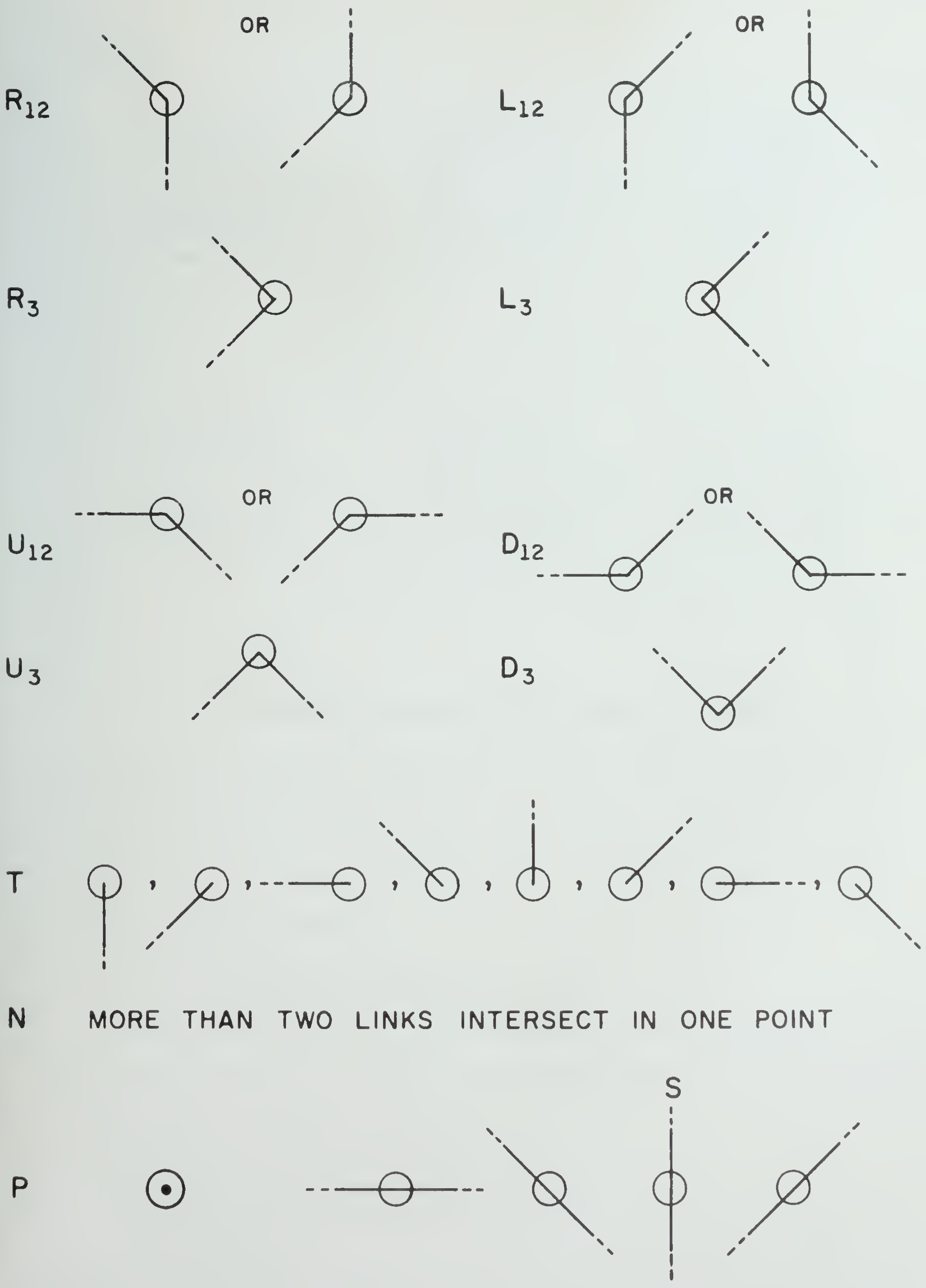


Figure 2.6 Line Representation of Node Types

$$R_{12}\ \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad L_{12}\ \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array} \qquad D_{12}\ \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$N\ \begin{array}{|c|c|c|}\hline 1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad U_{12}\ \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array} \qquad T\ \begin{array}{|c|c|c|}\hline 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Figure 2.7 Binary Representation of Nodes

b. Merging

Because of the way that people draw line-like patterns (section 1.4), the quantization noise and the small size of the pattern used in node extraction, one may expect that not all the extracted nodes are necessary for representing the drawn character. With this in mind it may be observed that a net change of line direction may be attributed to the sum of small local changes in direction and the place of this net change is not critical. One may also ignore changes in line direction (R, L, U, and D) when they are too close to line-intersections (N) or line-tips (T). To make use of these observations, line-like drawings were studied and a set of MERGING RULES were devised. These rules when applied to the NODEM contents leave some nodes as local features.

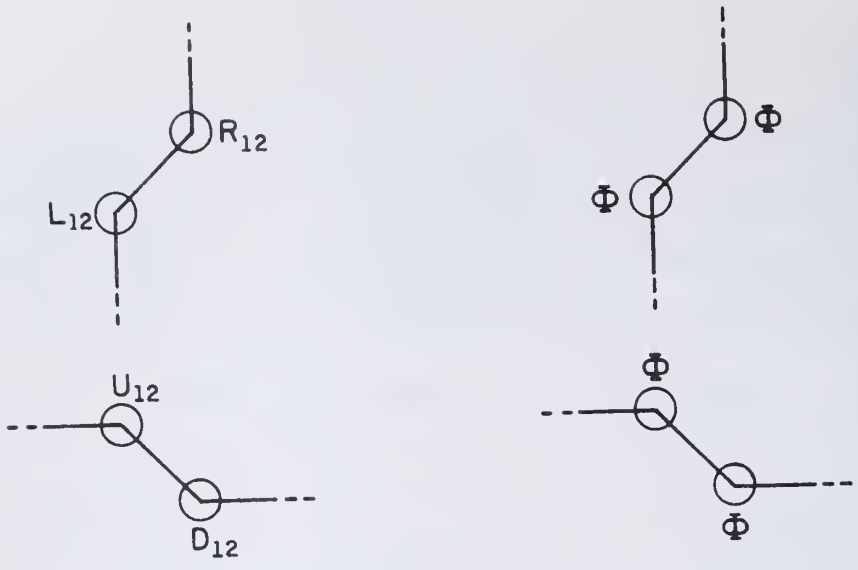
The MERGING RULES are designed to be applied on two neighboring nodal cells in a certain order defined later. Each two neighboring nodes can be seen as interacting and the result of interaction will determine the net result. Three types of interactions are defined between neighboring nodes:

i. Cancellation Interaction: (Figure 2.8(a))

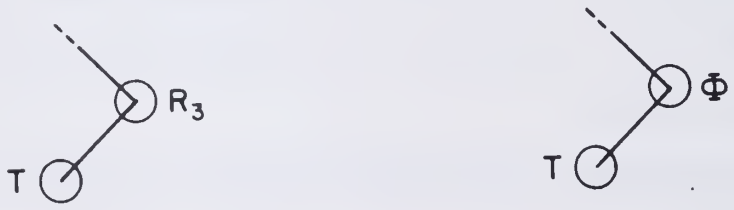
This occurs between R and L-type or U and D-type nodes. The result is to null the two nodes i.e., rewriting the two cells containing them as Ø type nodes.

ii. Domination Interaction: (Figure 2.8(b))

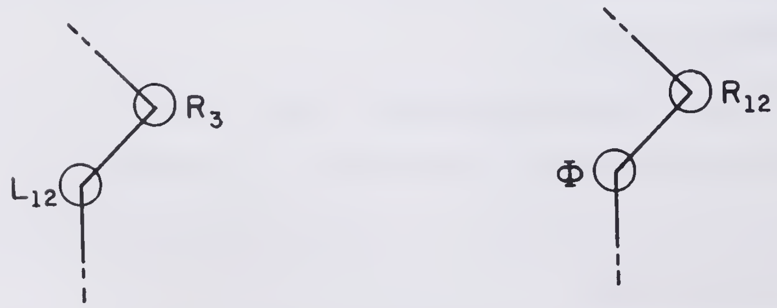
In this type of interaction one node dominates the other. After merging, the dominated node is nulled and the dominating node remains unaltered. Examples are interactions between N-type or T-type and



a) Cancellation Interaction



b) Domination Interaction



c) Modification Interaction

Figure 2.8 Examples of Nodal Interactions

any other type node resulting in dominating N-type or T-type nodes, respectively.

iii. Modification Interaction (Figure 2.8(c))

This is similar to the previous type except that the dominating node is rewritten as another node (modified) and the dominated node is deleted (nulled).

The ALGORITHM MERGE has been designed for performing the merging operation using the HSCANW23 scheme and applying the MERGING RULES (Figure 2.9).

ALGORITHM MERGE:

1. Scan NODEM using HSCANW23 (section 2.3).
2. For each position of W23 apply the ASCANN12 scheme.
3. For each position of W12 find the MERGING RULE which can be applied to merge the two cells seen through W12. Rewrite the contents of these two cells.

This algorithm needs five steps for each NODEM cell: one step for positioning W23 and four steps for scanning W23 using ASCANW12. Therefore, the total number of steps is $5N^2$, where N is the dimension of the NODEM matrix. However, as will be seen in the hardware implementation of the algorithm (section 3.6), one can reduce this number to $4N^2$ by overlapping the positioning of W23 while merging two cells.

2.7 Structural Features Extraction

Structurally, a character is described by means of horizontal (or vertical) segments connected together by links, as has been seen in Figure 2.1. This level of description will supress some of the

i. Cancellation Rules

$\emptyset\emptyset \leftarrow R_{12}L_{12}/L_{12}R_{12}/R_3L_3/L_3R_3$

$U_{12}D_{12}/D_{12}U_{12}/U_3D_3/D_3U_3$

ii. Domination Rules

$\left.\begin{array}{l} T\emptyset \leftarrow Tn \\ \emptyset T \leftarrow nT \end{array}\right\}$. where n is any node other than N.

$\left.\begin{array}{l} N\emptyset \leftarrow Nk \\ \emptyset N \leftarrow kN \end{array}\right\}$ where k is any node.

$\emptyset U_{12} \leftarrow R_{12}U_{12}/R_3U_{12}/L_{12}U_{12}/L_3U_{12}$

$U_{12}\emptyset \leftarrow U_{12}R_{12}/U_{12}R_3$

$U_{12}L_{12}/U_{12}L_3$

$\emptyset D_{12} \leftarrow R_{12}D_{12}/R_3D_{12}/$

$L_{12}D_{12}/L_3D_{12}$

$D_{12}\emptyset \leftarrow D_{12}R_{12}/D_{12}R_3/$

$D_{12}L_{12}/D_{12}L_3$

iii. Modification Rules

$\emptyset P \leftarrow TT$

$\emptyset R_{12} \leftarrow L_{12}R_3$

$R_{12}\emptyset \leftarrow R_3L_{12}$

$\emptyset L_{12} \leftarrow R_{12}L_3$

$L_{12}\emptyset \leftarrow L_3R_{12}$

$\emptyset U_{12} \leftarrow D_{12}U_3$

$U_{12}\emptyset \leftarrow U_3D_{12}$

$\emptyset D_{12} \leftarrow U_{12}D_3$

$D_{12}\emptyset \leftarrow D_3U_{12}$

Figure 2.9 MERGING RULES

details (local features) and can be used to classify the characters into subgroups having the same structures. It also allows for a variety of forms of the same character to be recognized using the structural features (Figure 2.10).

a. Segmentation

The character is segmented horizontally or vertically. A segment may consist of several separable subsegments if there are no lines interconnecting them within the segment. Each subsegment may consist of a part of a line or one or more local features. These local features are related within the subsegment by the spatial relationships 'left of' or 'above', in case of horizontal or vertical segmentaion, respectively.

Segmenting the character into horizontal segments will be considered here. Intuitively, the local features which appear in consecutive rows in NODEM could be included in the same horizontal segment. However, a limit on the number of these rows has to be set. Otherwise, a character which has local features in every row will be considered as a horizontal segment no matter what its size is. In this case the spatial interrelationship in the vertical direction will be lost. To avoid this difficulty and to be compatible with the previous processing (3x3 window), no more than two rows are merged into one row if each row contains at least one local feature. In other words the WORKING STORAGE memories (IMAGEM, LINKEM and NODEM) are segmented according to the local feature distribution in NODEM. In view of this discussion the following algorithm defines the boundaries of horizontal segments.

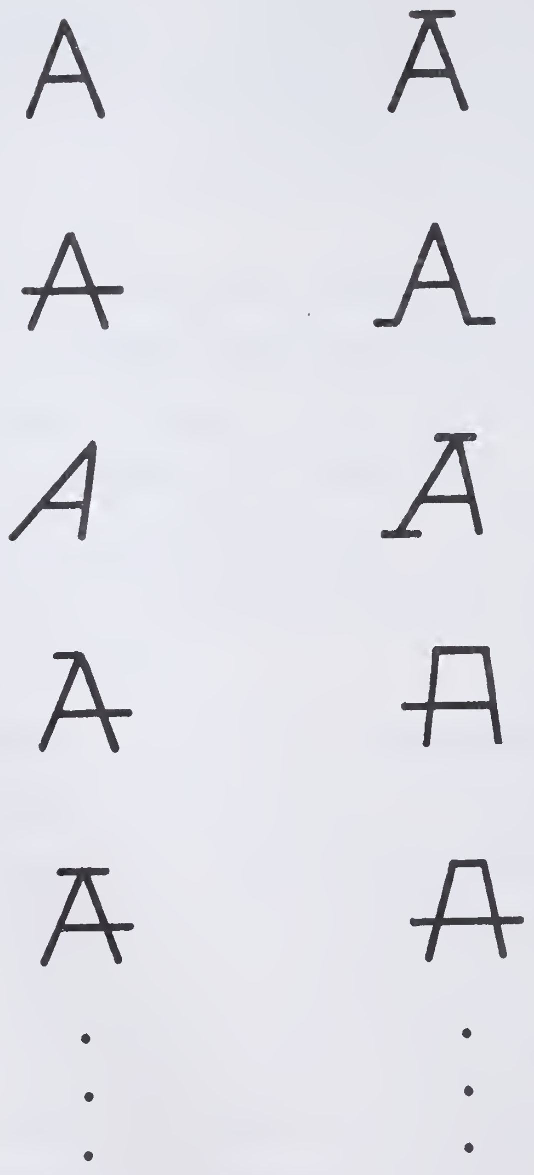


Figure 2.10 Different Shapes of 'A' with the Same Structural Features (Horizontal Segmentation)

ALGORITHM HSEGMENT:

1. The first segment begins at the first row of the working storage memories.
2. It ends at the row for which one of the following situations occurs:
   - i. Both the rows immediately above and below have local features.
   - ii. A previous row has local features in the same segment and the next row also has local features.
   - iii. The whole working storage has been scanned.
3. The next segment begins at the row immediately below the last segment.
4. It ends at the next row for which one of the previously mentioned situations (step 2) occurs.

While defining the boundary rows of a segment a vector is constructed to define the boundaries of its subsegments. This vector is called the separation vector (SEPVEC).

Construction of SEPVEC

SEPVEC has the same length as the dimension of the working storage memories (IMAGEM, NODEM and LINKM). It is constructed by projecting vertically the points (1's) contained in the segment onto a horizontal line (HSEPVEC) (Figure 2.11). In this example the segment under consideration consists of two subsegments. It is obvious that adjacent points (1's) in SEPVEC define the ranges of the subsegments separated by adjacent (0's).

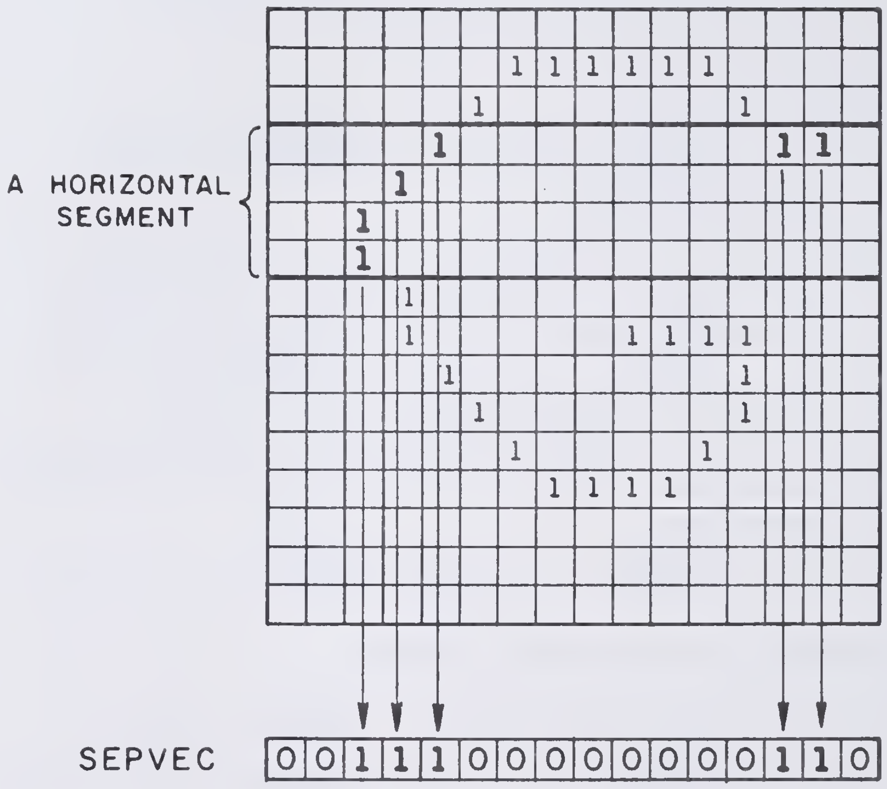


Figure 2.11 Example of Horizontal Separator Vector (HSEPVEC)

b. Linking of Segments

Linking is the property that two neighboring segments are connected by links. Since these links are at the boundaries of segments, they contain no nodal features. Links are designated as input (ILINKS) or output links (OLINKS) according to the way they connect the segment to its neighbors. ILINKS are either ULINKS or LLINKS. The OLINKS are either DLINKS or RLINKS according as the segmentation is horizontal or vertical (see Figure 2.1). In other words, IOLINKS flow vertically for horizontal segmentation--or horizontally for vertical segmentation. A simple way of extracting these IOLINKS is to eliminate horizontal or vertical lines from the binary image in IMAGEM for horizontal or vertical segmentation, respectively. The resulting binary pattern is stored in LINKM. LINKM is then segmented and the links that appear on the segment boundaries are the IOLINKS for the particular segmentation. An algorithm which will do just that follows.

ALGORITHM IOLINKS

1. Clear LINKM
2. Scan IMAGEM using the HSCANW12 (VSCANW12) scheme for horizontal (vertical) segmentation.
3. For each position of W12 in IMAGEM copy its right (down) cell into the corresponding cell of LINKM if and only if its other cell is 0.

After applying this algorithm, LINKM memory will contain a subset of the character binary image (IMAGEM). After segmentation if row I and row J ($I<J$) define boundary rows of a certain segment they will contain 1's equal to the number of ILINKS and OLINKS of that segment, respectively. These 1's will also define the points at which these links enter or leave the segment.

2.8 Feature Vector Construction

For each type of scanning a feature vector, HFETVEC or VFETVEC, is constructed. Each vector consists of three subvectors, two subvectors for the structural features and the third for local features. For HFETVEC the three subvectors are ULINKV, DLINKV and HTABV. Similarly VFETVEC consists of ILINKV, RLINKV and VTABV. ULINKV (DLINKV, LLINKV, RLINKV) is an ordered list of the codes of ULINKS (DLINKS, LLINKS, RLINKS) of the character segments, ordered according to the direction of scanning. HTABV (VTABV) is another subvector in which the distributions of local features within the subsegments of each character segment are coded.

Feature Vector Coding

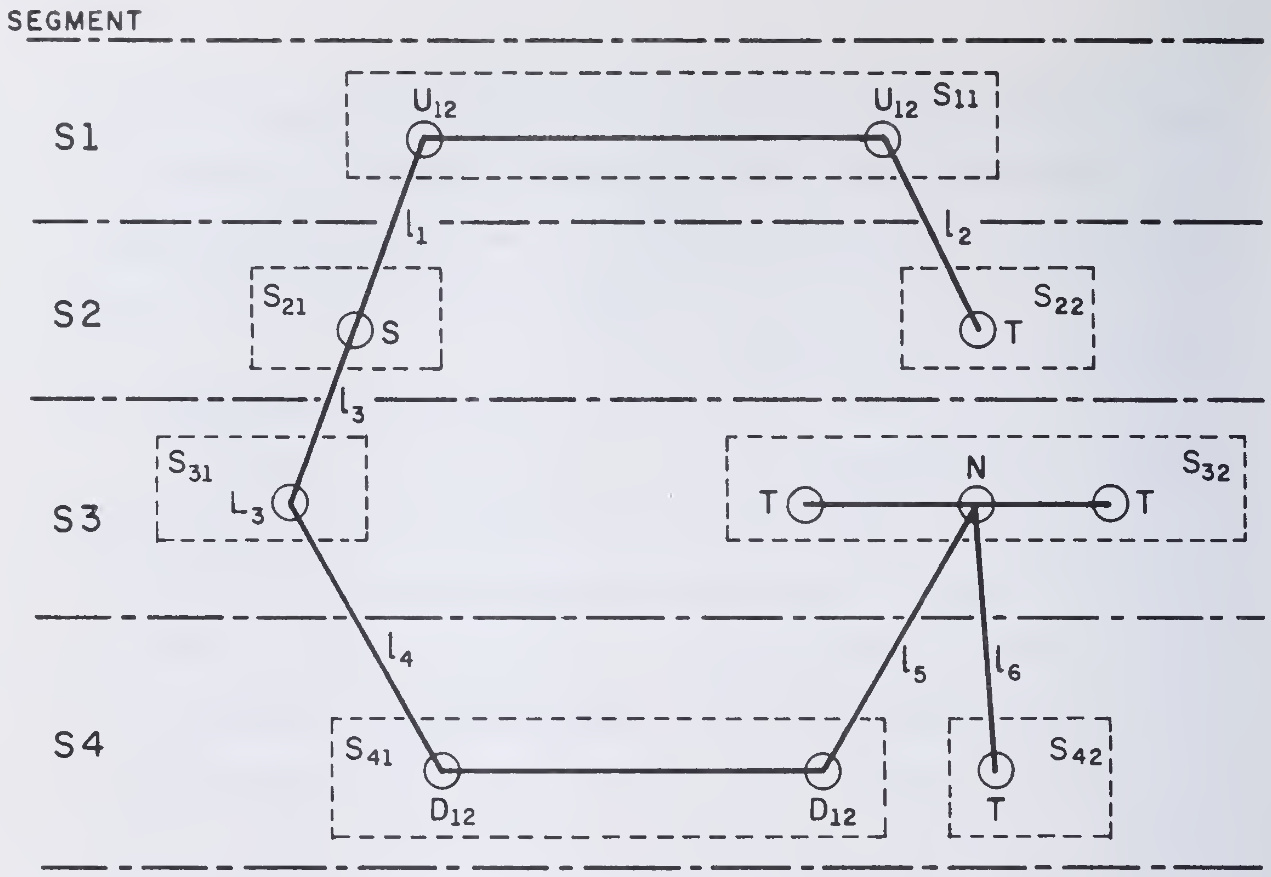
The case of horizontal scanning is considered here. We will assume that the WORKING STORAGE has been segmented horizontally. For each segment a separation vector (SEPVEC) has also been constructed which defines the subsegments boundaries. Initially, it is assumed that the subvectors of the feature vector contain 0's. It is also assumed that 4 is the limit on the number of features (LINKS or NODES) which can exist in a segment. Let each element (corresponding to a segment) in a subvector consists of 4 places which can accommodate the codes of these features. For each subvector there is a pointer and a counter. When a counter contains 0's, the corresponding pointer points immediately to the left of the 4 places reserved for the particular element. Then, the following coding algorithm is implemented.

FEATURE VECTOR CODING ALGORITHM (HFEVECAL)

1. Beginning with the top segment, scan each segment from left to right.
2. Count the number of features (LINKS or NODES) as the segment is scanned and store the result in the corresponding counter.
3. For each subvector the corresponding pointer is moved to the right the same number of places stored in their counters.
4. Insert '1' in the places indicated by the pointer.
5. After each segment is scanned the counters are set to 0 and the pointers point to their initial positions.

Examples (Figure 2.12)

To illustrate the construction of the feature subvectors resulting from horizontal scanning, it is assumed that the idealized representation of 'G' has been already segmented as shown in Figure 2.12(a). The heavy lines are the character image lines that are represented in binary form in IMAGEM. The small circles are the local features (nodes) which are represented by their codes in NODEM. The lines L_1-L_6 represent the LINKS which are stored in binary form in LINKM after eliminating all the horizontal lines for the character image. All other lines and symbols are for illustration only. Notice



a) Line Representation of 'G'

SEGMENT NO	1	2	3	4	
ULINKV	0 0 0 0	1 1 0 0	1 0 0 0	0 1 1 0	STRUCTURAL PARTS
DLINKV	0 1 0 0	1 0 0 0	1 0 1 0	0 0 0 0	
HTABV	0 1 0 0	1 1 0 0	1 0 0 1	0 1 1 0	LOCAL FEATURES PARTS
HNODEV	$U_{12} U_{12}$ Φ Φ	S T Φ Φ	L_3 T N T	$D_{12} D_{12}$ T Φ	

b) Feature Subvectors

Figure 2.12 Example of Feature Vector Coding

that there are no nodes on the boundaries between the character segments. One may also notice that there is no connection between subsegments within a segment. A fourth subvector, HNODEV is illustrated here (Figure 2.12(b)) to show the feasibility of adding a new level of detail in the feature vector. This subvector has not been included in the current implementation of INCOM.

The character is coded segment by segment beginning with the top one (S_1). Since this is a top segment ULINKV(1) will contain (0000). It cannot be divided into subsegments. The number of links are two, L_1 and L_2, which connect S_1 to the second segment, S_2. Therefore, DLINKV(1) is (0100). S_1 contains two nodes of the same type U_{12}. This fact is decoded as HTABV(1) and HNODEV(1) which are (0100) and ($U_{12}U_{12}$ØØ), respectively.

The second segment S_2 has two subsegments (S_{21},S_{22}) which contain two nodes a line (S) and tip (T). L_1 and L_2 link S_2 to S_1 while L_3 links S_{21} to the segment below (S_3). This information is coded into the subvectors as (1100), (1000), (1100) and (STØØ for ULINKV(2) DLINKV(2), HTABV(2) and HNODEV(2), respectively. Notice that the number of '1's in both ULINKV(2) and HTABV(2) is two. This indicates that S_2 consists of two subsegments, each contains one local feature (node) and each is connected by a link to the segment above. The third and fourth segments are coded similarly.

In the actual hardware implementation, the operations of segmentation, subsegmentation and coding the various subvectors are done concurrently in one WORKING STORAGE scan.

2.9 The CLASSIFIER

It consists of three parts, the V-ELEMENT CONSTRUCTION part, the DATA STRUCTURING part and the DECISION part (Figure 2.13). The classification operation can be rendered more efficient if it works on a set of features instead of only single features. Hence, compound features (V-ELEMENTS) are constructed from elements of the input features subvectors according to programmable schemes. These schemes are chosen to enhance the recognition rate and to reduce the number of training samples. A V-ELEMENT may appear in one or more characters. The relationship between an input FEATURE VECTOR, its constitutent V-ELEMENTS and the associated characters can be viewed as a tree (Figure 2.14). The root node is the input FEATURE VECTOR, the next level of nodes is the V-ELEMENTS and the terminal nodes (leaves) are the character codes. In the learning node this tree structure is embedded into the DATA STRUCTURING memories. For recognition this limited information about the character set is generalized to the testing set. A simple way of generalization is to use the number of leaves for each character code as the basis for decision; the code which scores the highest number is the recognized character.

There are two types of memories in the DATA STRUCTURING part; the FEATURE POINTER LIST (FEPOLIST) and the CONFUSION MATRIX (CONFUMAT). V-ELEMENTS are stored in FEPOLIST with associated pointers to words in CONFUMAT. Each CONFUMAT word specifies the character codes (leaves) which are associated with a particular V-ELEMENT. To economize in memory and to speed up the recognition process, the pointer part of a FEPOLIST word contains a character code if it is unique to that V-ELEMENT. The following two algorithms are used for learning and recognition, respectively.

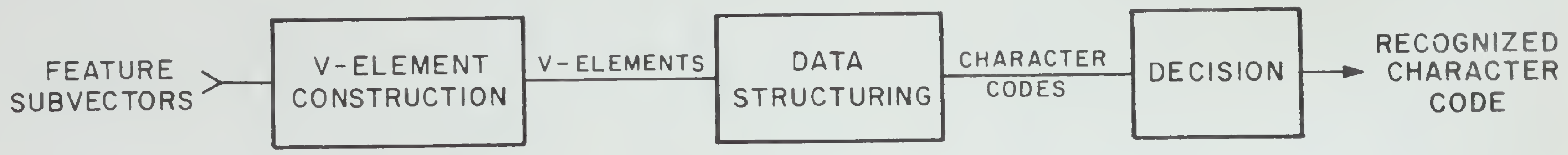


Figure 2.13 INCOM CLASSIFIER Block Diagram

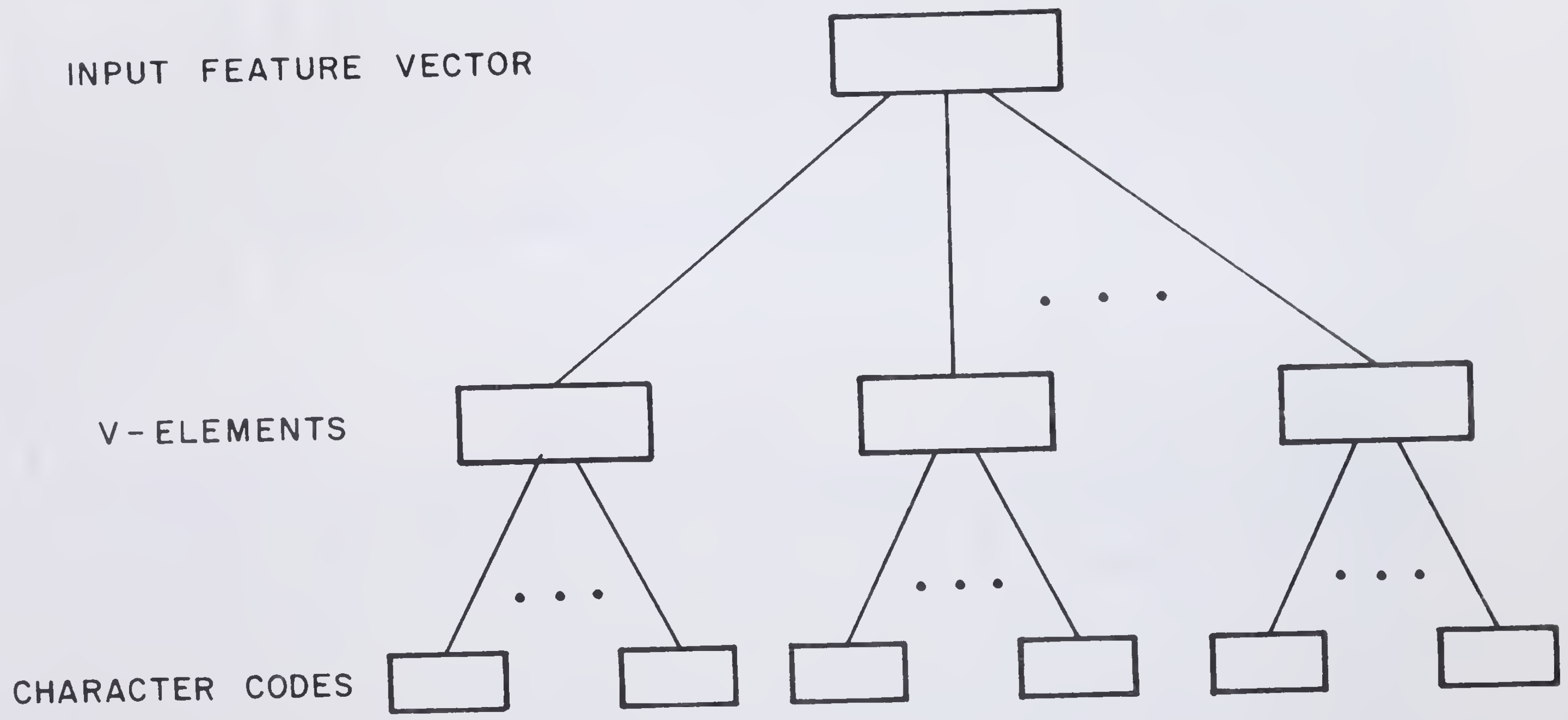


Figure 2.14 The Relationship between the Feature Vector, the V-ELEMENTS and the Character Codes

ALGORITHM LEARN:

1. Construct a V-ELEMENT from the input subvectors.
2. Search FEPOLIST to find this V-ELEMENT. If it is found go to Step 4.
3. Store the constructed V-ELEMENT in a new FEPOLIST word with the pointer part containing the input character code.
4. If the V-ELEMENT is unique to this code go to Step 5. Otherwise, one of two possibilities may arise. If the pointer part contains a character code not equal to the input code, both codes will be stored in a new CONFUMAT word and the pointer rewritten to contain the address of this word. If the pointer part addresses a CONFUMAT word the new input code is simply added to this word.
5. Repeat the above steps for all possible V-ELEMENTS.

ALGORITHM RECOGNIZE:

1. Construct a V-ELEMENT.
2. Search FEPOLIST to find this V-ELEMENT. If it is found the pointer part is checked if it indicates a character code. If it is a character code this will be displayed as the recognized code and the recognition process is terminated. Otherwise, the pointer addresses a CONFUMAT word which is read and sent to the DECISION module. On the other hand if this V-ELEMENT is not found go back to Step 1 for a new V-ELEMENT.
3. The DECISION module computes the frequency of occurrence of the character codes in the CONFUMAT words and displays the character code which scores the maximum frequency as the recognized character.

2.10 V-ELEMENT CONSTRUCTION

The V-ELEMENT CONSTRUCTION module constructs V-ELEMENTS from the input FEATURE VECTOR memories according to a set of construction schemes resident in a writable memory (VMEM). These construction schemes can be chosen such that the constructed V-ELEMENTS enhance the recognition rate of the classifier. This flexibility in the design makes it possible to tailor several schemes for several character sets, if necessary. All the resident schemes in VMEM are used to construct the V-ELEMENTS when building the classification dictionary in the learning mode. However, it is not necessary to try all of them in the recognition mode. They are ordered hierarchically according to the amount of detail in the information content which will construct a V-ELEMENT. This means that the structural features parts of the FEATURE VECTOR, e.g. ILINKV and OLINKV, are tried first followed by the more detailed information (local features) (TABV). This order results in faster recognition response.

Some construction schemes may combine features from different segments of the character. This results in constructing V-ELEMENTS which are rich in structural information. A simple, but powerful, combination can be obtained by looking at the character from two opposite directions; top and bottom or left and right for horizontal or vertical scanning, respectively. Since this is independent of the character length (number of segments) it can also be regarded as a kind of normalization on the FEATURE VECTOR level. This helps in recognizing different shapes of characters of different lengths and in reducing the number of confused characters (Figure 2.15). The

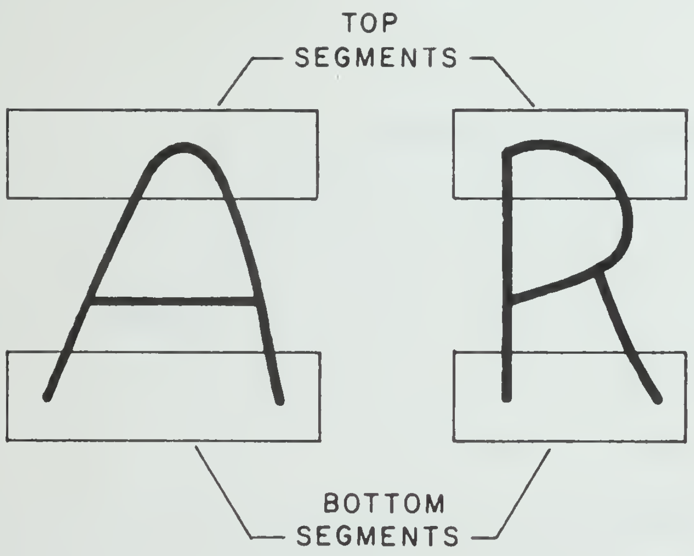


Figure 2.15 Characters with the same Top and Bottom Segments

result is faster processing and better recognition rate.

A V-ELEMENT consists of an identification part and a feature part (Figure 2.16(a)). The identification part contains the name of the construction scheme used, a position tag which indicates the positions (segment numbers) of the feature(s) in the FEATURE VECTOR and the mode of scanning. The feature part consists of the codes of these features.

2.11 DATA STRUCTURING and LEARNING

The DATA STRUCTURING part contains two memories: the FEATURE POINTER LIST (FEPOLIST) and the CONFUSION MATRIX (CONFUMAT). A FEPOLIST word is divided into three parts: a feature (F-part), a pointer (P-part) and a type of pointer bit (TP) (Figure 2.16(b)). The F-part contains a V-ELEMENT and the P-part is interpreted according to the TP bit. When TP is '1' the P-part corresponds to a character code and the F-part contains a V-ELEMENT which is unique to this character. When the F-part contains a V-ELEMENT which had appeared in several input FEATURE VECTORS while learning, TP will be '0' and the P-part will contain a pointer to a word in CONFUMAT. This word holds a representation of the character codes associated with this V-ELEMENT. Each bit position of a CONFUMAT word corresponds to a character code (Figure 2.16(c)). It contains '1' only when a P-part of the FEPOLIST word is addressing this CONFUMAT word and the F-part contains a V-ELEMENT associated with the corresponding character code.

The FEPOLIST can be constructed as an associative memory which can be searched by the contents of its F-parts. This facilitates the

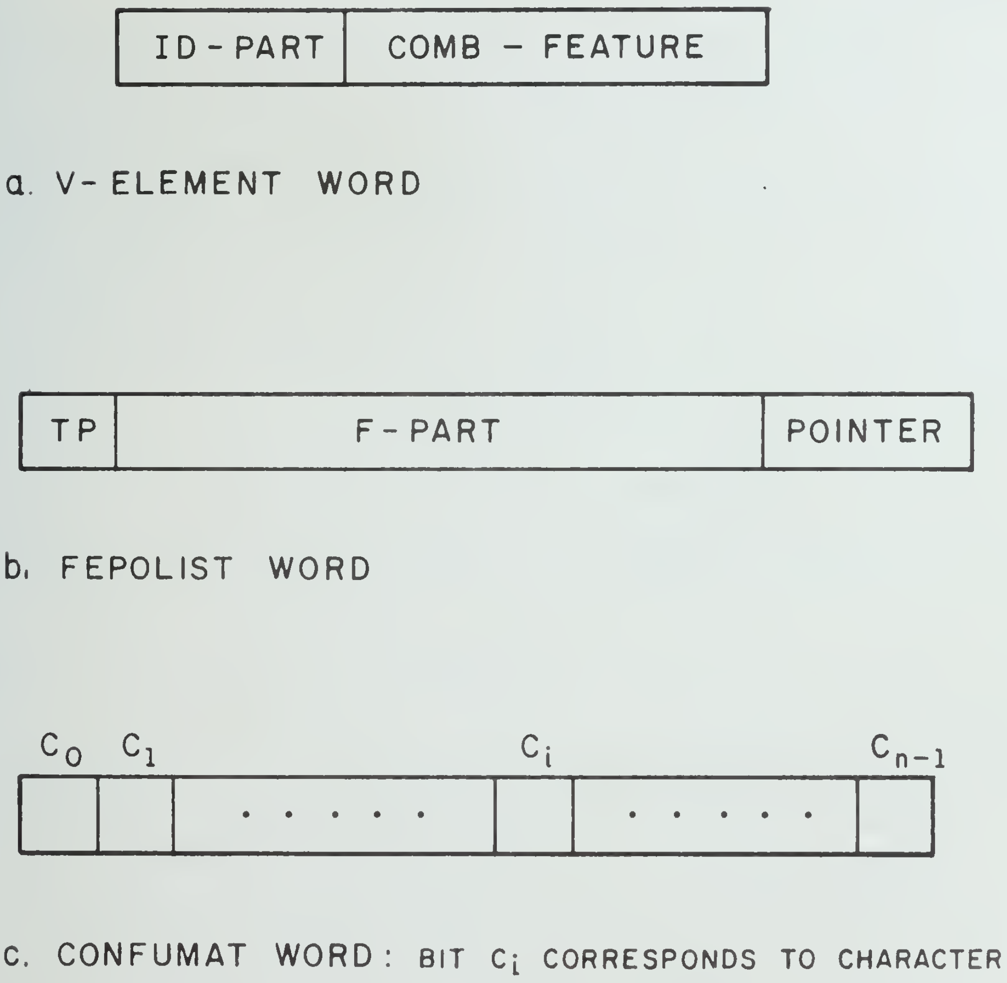

ID-PART
COMB - FEATURE
a. V-ELEMENT WORD
TP
F-PART
POINTER
b. FEPOLIST WORD
C_0 C_1 C_i C_{n-1}
c. CONFUMAT WORD: BIT C_i CORRESPONDS TO CHARACTER

Figure 2.16 CLASSIFIER Word Partitions

expansion of the memory word-wise to store more V-ELEMENTS or bit-wise when changing the length of its main parts.

The CONFUMAT memory can be expanded horizontally for more character codes and vertically when different character confusion combinations (words) are needed. For more sophisticated processing, one can make use of the possibility of finding words of similar 1's distribution. In this case one word is enough to represent this character combination and accordingly the associated P-parts of FEPOLIST are changed to the new word location.

The DATA STRUCTURING module works in two modes. In the learning mode, a new V-ELEMENT is stored in the F-part of an available word in FEPOLIST, the TP bit is set to '1' and the input character code is stored in the P-part. When FEPOLIST is searched and the V-ELEMENT is found several possibilities may arise. When TP=0 (the P-part contains a pointer to a word in CONFUMAT) this word is read and stored in the CONFUMAT buffer. A '1' is inserted in the buffer in the bit location corresponding to the input character code. Then, the new word is rewritten in the same addressed location. When TP=1 and the input character code is equal to the contents of the P-part further processing is not needed. Otherwise a new CONFUMAT word is created in which the bit positions corresponding to both the codes of the P-part and the input character are set to '1' and its location is written in the P-parts. The TP bit is reset to '0' to indicate that the new P-part contains a pointer to a CONFUMAT word.

2.12 DECISION and RECOGNITION

In the recognition mode, the FEPOLIST memory is searched for a match between its F-part and each newly constructed V-ELEMENT (Figure 2.17). If no match is found the search continues for another V-ELEMENT. On the other hand, if a match is found, the TP bit of the matched FEPOLIST word is read as well as its P-part for further processing. If the TP bit contains '1', then the P-part contains a character code and the recognition cycle terminates by displaying this code as the recognized character. However, if TP contains a '0' (the P-part contains the address of a CONFUMAT word) this means that more than one character code shares the matched V-ELEMENT. The addressed CONFUMAT word is read and sent to the DECISION module.

The DECISION module contains a set of registers which contain the accumulated additions (scores) of the words fetched from CONFUMAT. In the current implementation, the character code which scores the highest number of 1's in the total number of fetched words is considered the recognized character.

The recognition cycle (Figure 2.17) is terminated when either a unique character code has been found (TP=1) or when all the V-ELEMENTS have been processed.

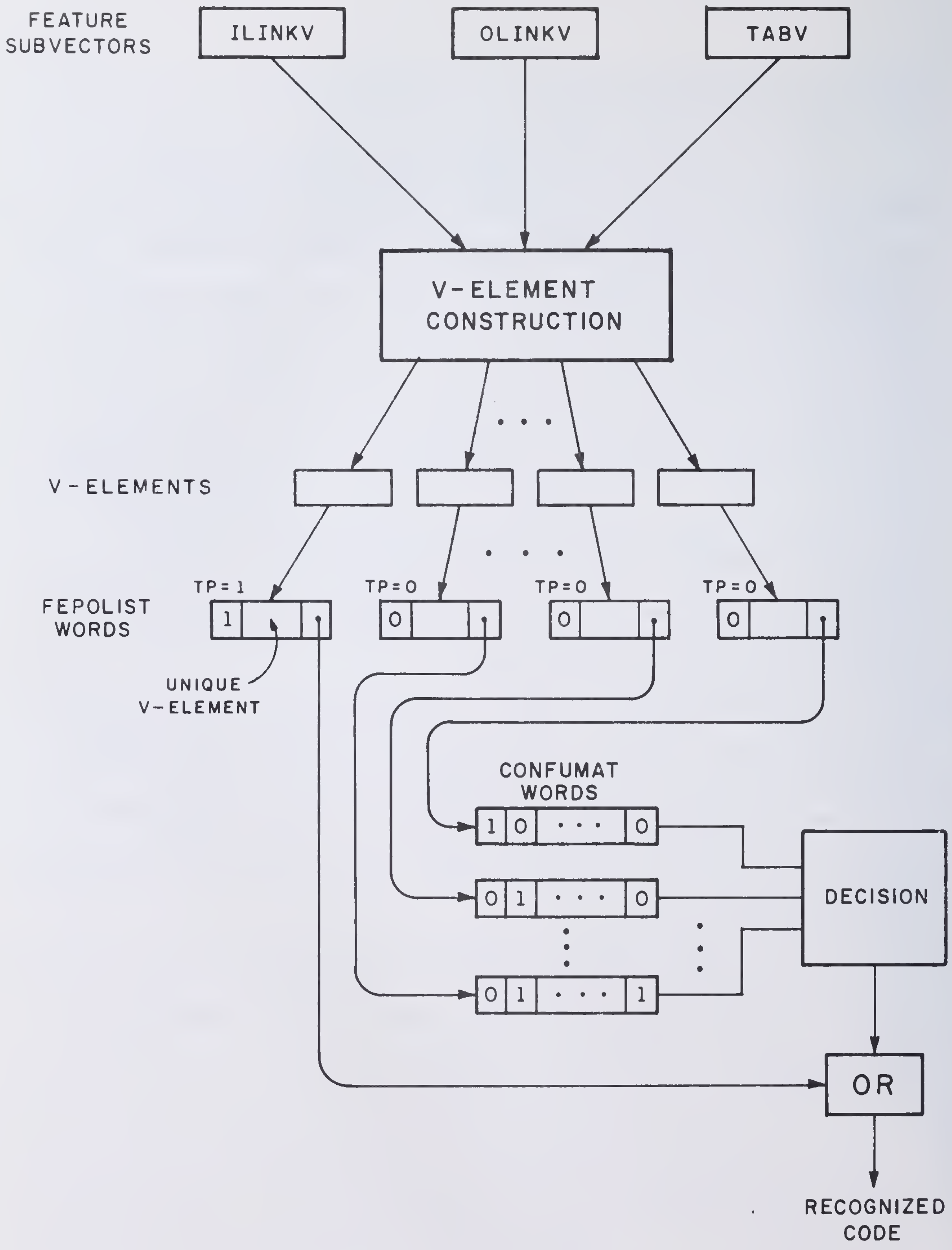


Figure 2.17 Recognition Cycle

3. HARDWARE IMPLEMENTATION

3.1 General Discussion

The system consists functionally of the PANEL and three processors: the PREPROCESSOR, the FEATURE EXTRACTOR and the CLASSIFIER (Figure 3.1). The user interacts with INCOM through the PANEL switches and indicators. The three processors constitute the pattern recognition part of INCOM and have been designed to implement the procedures described in Chapter 2. Architecturally, the system can be viewed as consisting of three sets of modules; the memory modules, the data manipulation modules and the control modules (Figure 3.2). Each one of the processors consists of several of these modules. The memory modules hold the information belonging to the processors. Some memory modules are shared, or operated on, by more than one processor, e.g., the WORKING STORAGE is shared by the PREPROCESSOR and the FEATURE EXTRACTOR. The data manipulation modules perform the operations on the contents of the memory modules. The memory modules communicate through the data manipulation modules. The third set of modules, the control modules direct the operations of the other modules and supervise the transfer of information between them.

A particular control module, the MASTER control (M-CONTROL), supervises the sequencing of the other control modules. It also handles direct control functions on some of the data manipulation and memory modules when the controls are simple and no separate modules

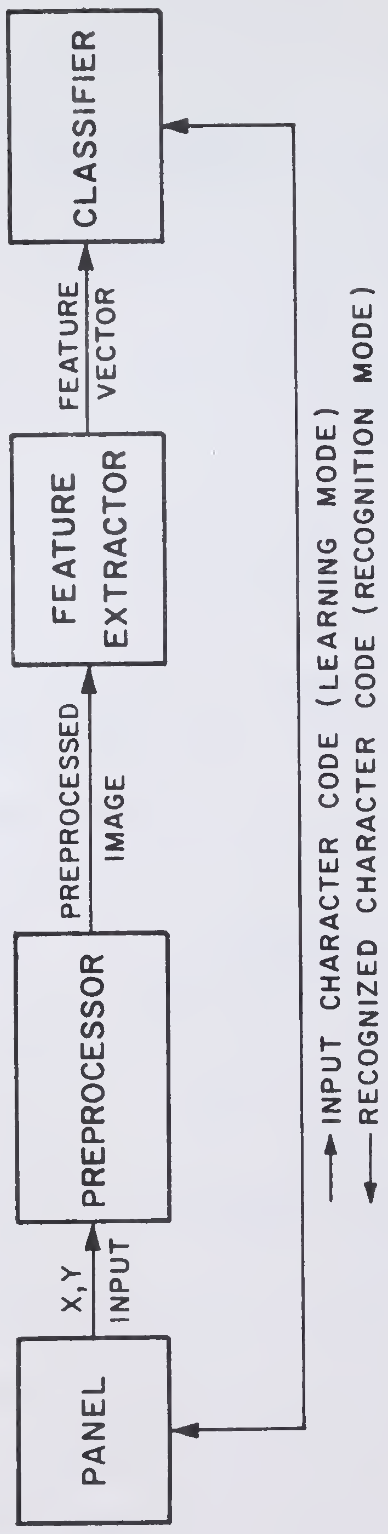


Figure 3.1 INCOM General Block Diagram

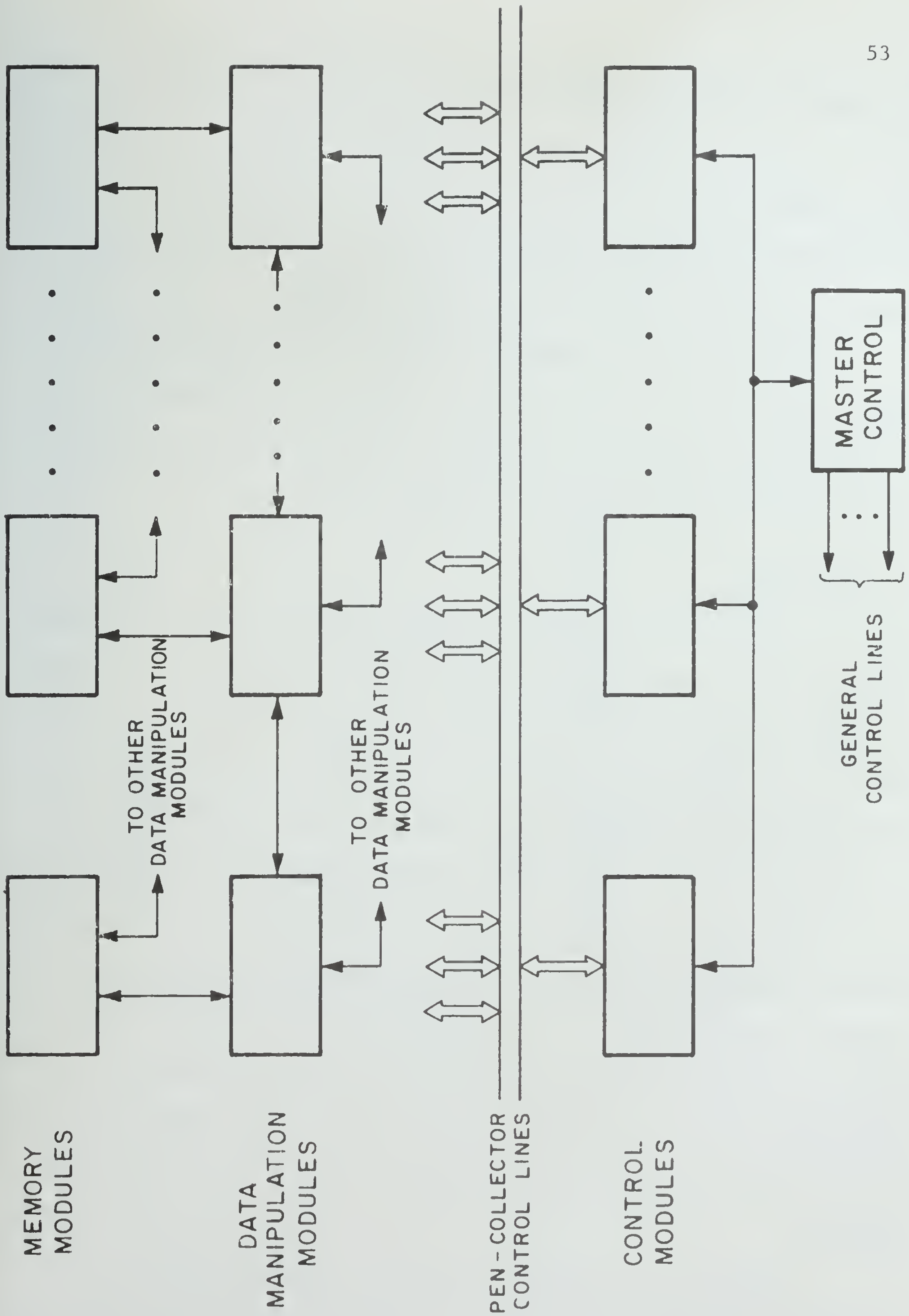

MEMORY MODULES
DATA MANIPULATION MODULES
OPEN-COLLECTOR CONTROL LINES
CONTROL MODULES
TO OTHER DATA MANIPULATION MODULES
TO OTHER DATA MANIPULATION MODULES
MASTER CONTROL
GENERAL CONTROL LINES

Figure 3.2 INCOM Architecture

are needed to perform them.

A control module produces open-collector signals which can be wire-ORed with other control signals. This arrangement facilitates modularizing the controls as well as enabling the addition and modification of control signals or even entire modules.

In terms of the algorithms described in Chapter 2, the memory modules hold the data structures inherent in the algorithms, the data manipulation modules perform the operations and the control modules implement the sequencing of these operations.

3.2 The PANEL

Besides its communication role as input-output tool the PANEL (Figure 3.3) has been designed to help the user to direct the operation of INCOM. It also has debugging aids which help in constructing the device. Its switches and indicators are grouped into OPERATION SELECTION, CLOCK, SIMULATED TABLET SIGNALS, CONTROL COUNTERS, DATA IN and DATA OUT groups. The OPERATION SELECTION group switches are set according to the chosen machine mode. The MODE switch is active only when the OPERATION SELECTION switch is in the NORMAL position. By choosing the RECOGNITION or the LEARNING position of the MODE switch the user can make the machine perform these respective functions. In the PROGRAM position the INPUT switch is active and the DATA IN switches are interpreted by the machine according to the position of the INPUT switches.

To help in operating the machine, status LEDs are lit when actions are required from the user. To inform the user the PROCEED LED

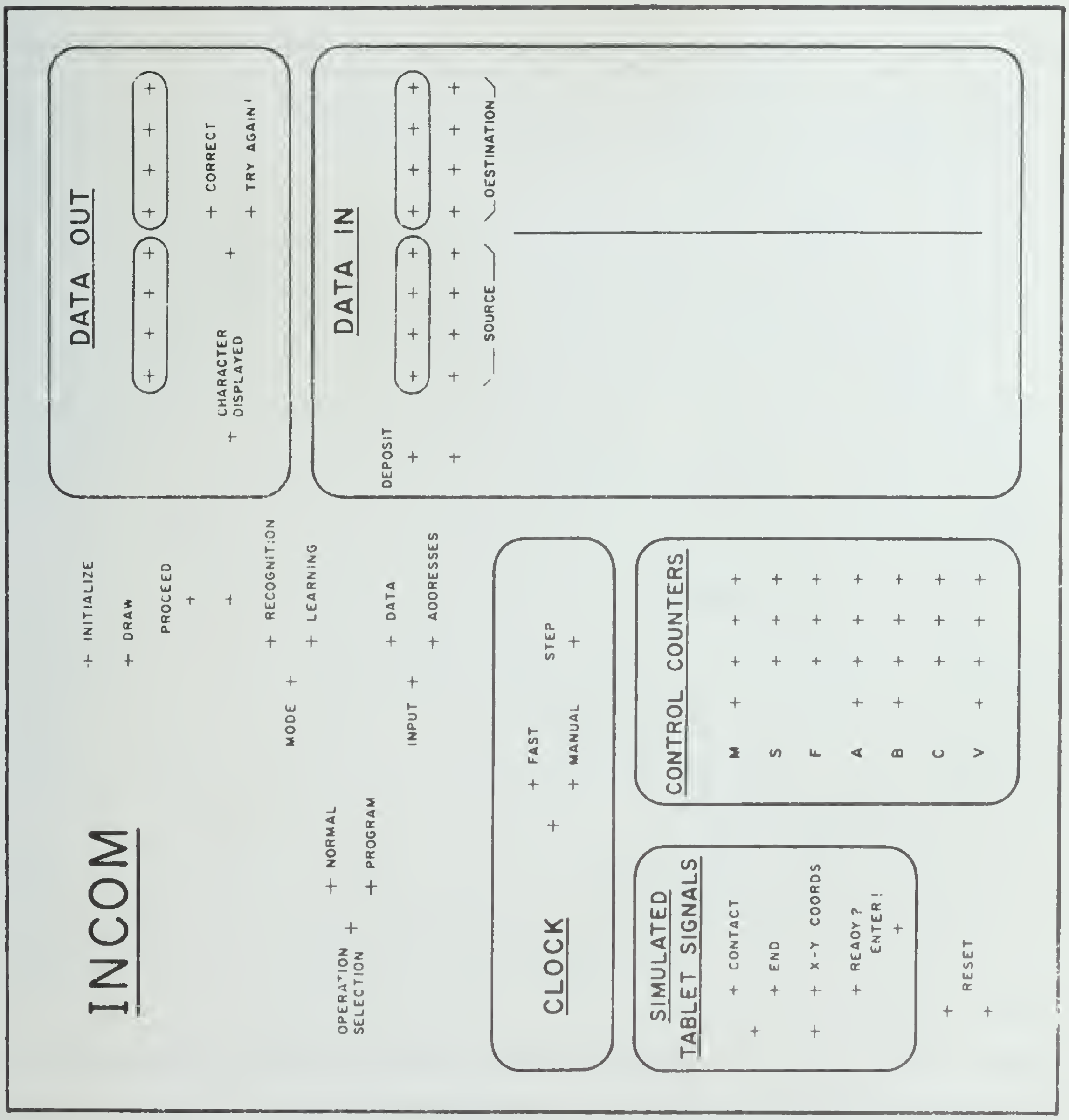


Figure 3.3 The PANEL

is turned on in conjunction with one or more of the remaining status LEDs. These are INITIALIZE and DRAW in the OPERATION SELECTION group, the CHARACTER DISPLAYED in the DATA OUT group and the DESPOSIT LED in the DATA IN group. The user should act according to the requirements of the status of the machine before pressing the PROCEED button. Once this button is pressed the machine resumes operation until it reaches another state in which a status LED is lit asking for a new action from the user. When the INITIALIZE LED is on, the machine is in the INITIALIZATION state. If the user wants to train the machine from scratch, the main memories of the machine are initialized. If this is not the case and the operation is NORMAL, the MODE switch is positioned accordingly. In the LEARNING mode the input character code is inserted via the DATA IN switches before pressing the PROCEED button. In the RECOGNITION mode there is no need to input anything at this stage, except to push the PROCEED button. This will make the machine go to the next state, DRAW, and the DRAW LED is lit. The user can now draw a character point by point by pressing the ENTER Switch. The DRAW action using the SIMULATED TABLE SIGNALS is enabled when these switches are in the CONTACT and X-Y COORDS positions. When the user has completed the drawing, the upper switch must be placed in the END position for the machine to begin processing the stored image of the drawn character.

As a debugging aid the master CLOCK can be FAST or MANUAL. In the FAST mode the CLOCK runs at its maximum speed, while in the MANUAL mode pressing the STEP button will pass one CLOCK pulse only. This is used for single stepping through the machine states. The machine states are displayed using the CONTROL COUNTERS' LEDs. Each control counter is indicated by a letter (M,S,F,A,B,C,V) followed by corresponding LEDs.

The DATA IN group consists of the DATA IN switches and LEDs and the DEPOSIT switch and its LED. The DATA IN switches are enabled when the MODE switch is in the LEARNING position or the OPERATION SELECTION switch is in the PROGRAM position. Information to be input is set up on the DATA IN switch and then the DEPOSIT switch is pressed.

The DATA OUT group consists of the DATA OUT LEDs, which display the code of the recognized character, the CHARACTER DISPLAYED status LED, and the CORRECT/TRY AGAIN! switch. The CHARACTER DISPLAYED LED is on when the machine is in the recognition mode and asking the user if satisfied with the character code displayed in the DATA OUT LEDs. Accordingly, the user will set the switch to CORRECT if satisfied or to TRY AGAIN! otherwise. The PROCEED switch is then pressed to let the machine continue operation.

3.3 System Organization and the MASTER CONTROL

Before describing in detail the different processors and modules of INCOM, the operation of INCOM will be described step by step following the flow chart of the MASTER CONTROL (M-CONTROL) (Figure 3.4). This description will provide a general understanding of the different processes involved in learning and recognition of input characters.

Reference is also made to the SYSTEM DATA FLOW DIAGRAM (Figure 3.5) in conjunction with the description of the operation of the M-CONTROL. For convenience Table 3.1 of PROCESSORS-OPERATIONS-MODULES is included, in which the different operations performed in

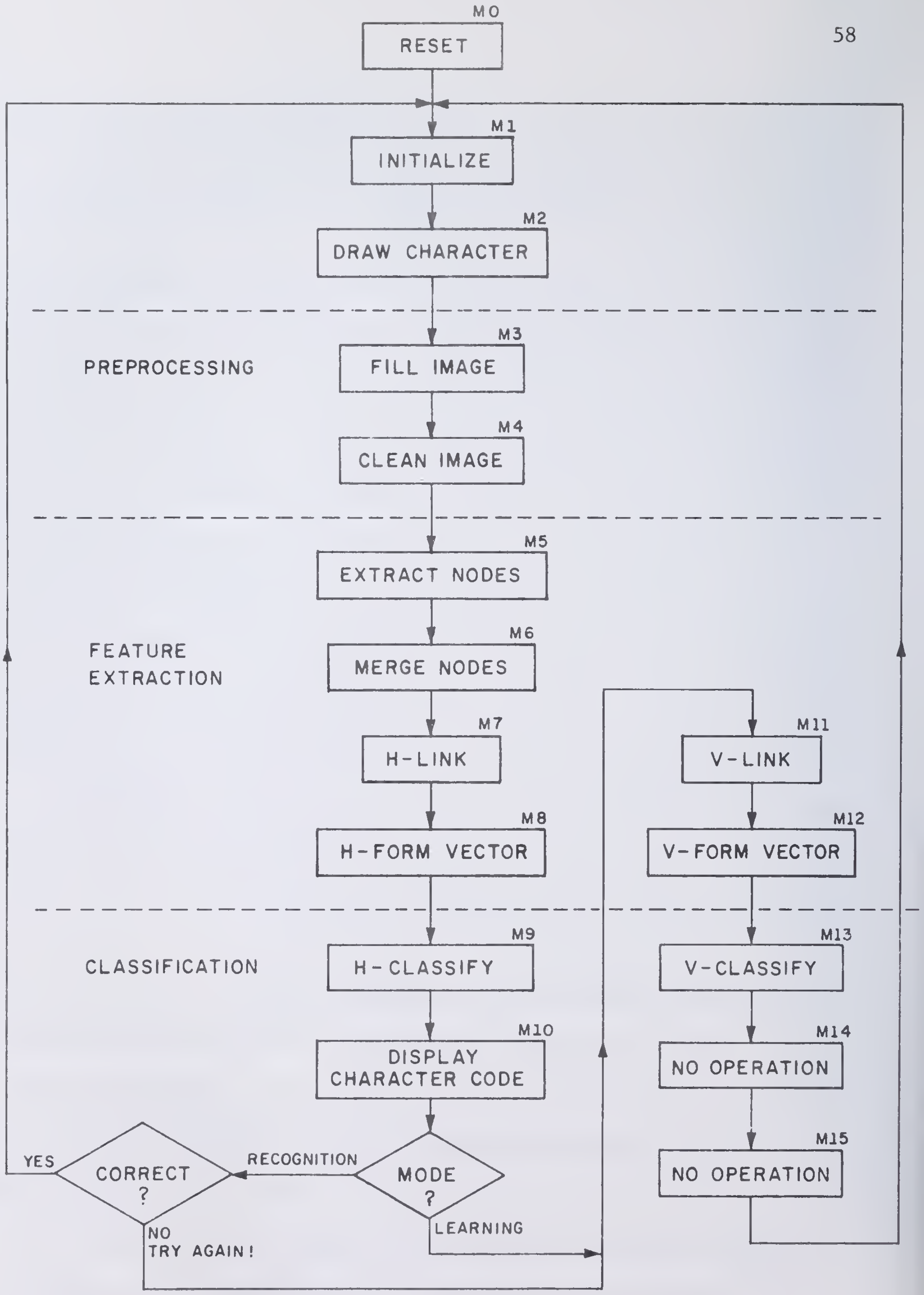


Figure 3.4 System Operation Flow Chart (M-CONTROL)

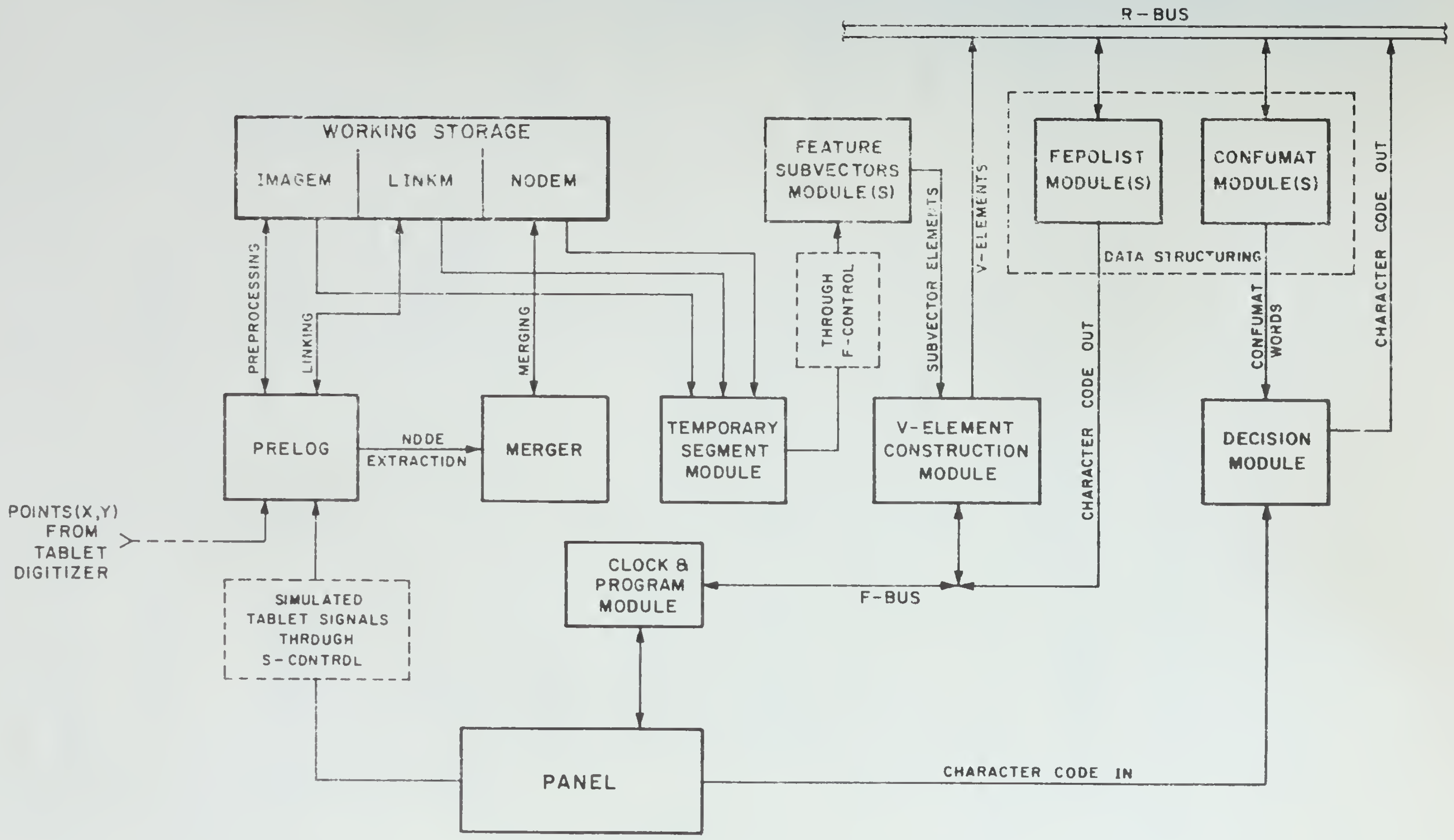


Figure 3.5 System Data Flow Diagram

Table 3.1 PROCESSORS-OPERATIONS-MODULES

PROCESSORS	SUBPROCESSOR/ PARTS	OPERATIONS PERFORMED	MODULES INVOLVED			
			MAIN CONTROLS	SPECIAL CONTROLS	DATA MANIPULATORS	MEMORY MODULES
PREPROCESSOR		-DRAW -FILL -CLEAN	M-CONTROL	S-CONTROL	PANEL+ PRELOG PRELOG	IMAGEM
FEATURE EXTRACTOR		-NODE EXTRACTION -MERGING -LINKING	M-CONTROL	-SEQUENCER	-PRELOG -MERGER -PRELOG	-IMAGEM+NODEM -NODEM -IMAGEM+LINKM
		-SEGMENTATION -FEATURE VECTOR FORMATION	F-CONTROL	——	-TEMP, SEGM. -ENCODING LOGIC	-IMAGEM+NODEM +LINKM -SUBVECTORS MEMORIES
CLASSIFIER	DATA STRUCTURING CONSTRUCTIO	-V.EL. CONSTRUCTION	A-CONTROL	V-CONTROL	-V-ELEMENT CONSTRUCTION MODULE	-SUBVECTORS MEMORIES RF resistor OF FEPOLIST
		LEARNING/ RECOGNITION	A-CONTROL/ B-CONTROL	——	-INSERT LOGIC OF CONFUMAT	FEPOLIST+ CONFUMAT+ CHAR res. of DECISION MODULE+ CHAROUT of PANEL
	DECISION	RECOGNITION	A-CONTROL	C-CONTROL	-DECISION MODULE	RF of FEPOLIST

INCOM are listed in conjunction with their associated modules.

The M-CONTROL has 16 states (M0-M15), two of which, M14 and M15, are dummy states in which no operations are performed. The normal learning cycle begins with M1, ends in M13 and returns to M1. In the first phase of recognition, M2-M9, the character image is processed by scanning it horizontally, while in the second phase M11-M13, the scanning is done vertically. The code of the recognized character is displayed in M10 after the first phase, waiting for the user response. According to the response, the machine may repeat the first phase by returning to M1 and waiting for a new character. Otherwise, the machine will start the second phase completing the process, displaying the character code and returning to M1. While describing the operations of INCOM the actions required by the user will also be stated.

M0: RESET

On power on, or when the RESET switch is pressed, the M-CONTROL counter is cleared and the RESET LED is turned on. The WORKING STORAGE memories (IMAGEM, NODEM and IOLINKM) are also cleared in this state. The PROCEED LED will be ON waiting for the user to press the PROCEED switch. When pressed, the machine will proceed to the next state, M1.

M1: INITIALIZE

The INITIALIZE and PROCEED LEDs will be on indicating that the machine is ready for initialization. If a new set of V-ELEMENT CONSTRUCTION SCHEMES are to be written into the VMEM memory of the

V-ELEMENT CONSTRUCTION MODULES, the OPERATION SELECTION switch is set in the PROGRAM position and the user enters the necessary information using the DATA IN group and INPUT mode switch. Each time a new set of these SCHEMES is introduced the CLASSIFIER memories (FEPOLIST and CONFUMAT) have to be initialized. This initalization involves clearing CONFUMAT and FEPOLIST memories.

In the normal operation of INCOM the above initialization is done only once, during start-up. After that the OPERATION SELECTION switch will remain in the NORMAL position while the MODE is set either in the LEARNING or the RECOGNITION position. In the LEARNING mode the input character code is deposited in the CHAR register of the DECISION module using the DATA IN group. This will place the character code into the CHAR register/counter of the DECISION module.
In the recognition mode the DATA IN switches are disabled, and this step is ignored. In either case PROCEED has to be pressed.

M2: DRAW CHARACTER

The DRAW LED will be on signalling the user to begin drawing a character. The CONTACT/END switch, of the SIMULATED TABLET SIGNALS group, should be in the CONTACT position as long as the user has not finished drawing the input character. When the user flips the switch to the END position the machine proceeds to the next state and begins processing the character. This DRAW operation is under the supervision of the S-CONTROL (Appendix A).

M3: FILL IMAGEM. M4: CLEAN IMAGEM

The preprocessing operations of Fill-in-gap and Clean-image are performed by the PRELOG module under the supervision of the M-CONTROL while scanning IMAGEM horizontally. Upon receiving a completion of scanning signal from the PRELOG module the machine proceeds to the next state enabling the node extraction operation.

M5: EXTRACTION NODES

This is the front end state of the FEATURE EXTRACTION processor. While scanning both IMAGEM and the node memory, NODEM, nodes (MNEX) are extracted from IMAGEM by the PRELOG module. They are routed through the MERGER module to store them in their corresponding cells in NODEM. At the end of scan ($\overline{\text{CRV}}=0$) the machine proceeds to MERGE.

M6: MERGE

During the MERGE cycle, the COPYING REGISTERS in the MERGER module will act as the W23 window of NODEM. While scanning NODEM, the COPYING REGISTERS contents are merged using the MERGING RULES and shifted back into NODEM. One extended scanning cycle is needed for the completion of merging. This cycle needs 4x256=1024 Master clock cycles.

M7: H-LINK, M11: V-LINK

While scanning the WORKING STORAGE the LINKING part of the PRELOG module extracts LINKS from IMAGEM and stores them in the LINKM memory. The H-LINK and the V-LINK operations are identical in every respect except for the scan mode.

M8: H-FORM VECTOR, M12: V-FORM VECTOR

The FEATURE VECTOR FORMATION SUBPROCESSOR control (F-CONTROL) is enabled while scanning the WORKING STORAGE memories. The SUB-VECTORS, ILINKV, OLINKV and TABV, are formed using the TEMPORARY SEGMENT module, the F-CONTROL and the ENCODING LOGIC of the FEATURE SUBVECTORS module. They will be stored in their respective memories of the last module. According to the direction of scanning the H-FORM or the V-FORM is enabled. One scanning cycle is all that is needed to complete the FORM operation. At the end of scanning, the F-CONTROL returns control to the M-CONTROL which proceeds to the next state.

M9: H-CLASSIFY, M13 V-CLASSIFY

The CLASSIFIER processor is enabled in this state. It consists of the V-ELEMENT CONSTRUCTION, the DATA STRUCTURING and the DECISION parts. The first, with its control (V-CONTROL), constructs the front-end of the CLASSIFIER. It reads parts of the SUBVECTORS memories forming a V-ELEMENT. Each formed V-ELEMENT will be in the register file RF of the FEPOLIST modules of the DATA STRUCTURING part waiting for further processing. The DATA STRUCTURING part has two control modules: the A-CONTROL and the B-CONTROL. The A-CONTROL acts as the main control of the CLASSIFIER processor while the B-CONTROL complements the operation of the A-CONTROL.

In the learning mode new entries are added to, or modifications are performed on the contents of the memories of the DATA STRUCTURING SUBPROCESSOR (FEPOLIST and CONFUMAT) to accommodate the input character code and its constructed V-ELEMENTS. When all the possible V-ELEMENTS of the current scanning mode have been processed the A-CONTROL returns control to the M-CONTROL.

In the recognition mode the FEPOLIST memory is searched and the CONFUMAT memory is retrieved to find the character codes corresponding to the V-ELEMENTS constructed. If a V-ELEMENT is found in FEPOLIST which matches the constructed V-ELEMENT of the drawn character and has a unique character code associated with it, this character code will be displayed in the DATA OUT group of the PANEL. Then control will return to the M-CONTROL. On the other hand, if a V-ELEMENT is found with a pointer to a word in CONFUMAT, this indicates that there is more than one character associated with this V-ELEMENT. The CONFUMAT is read, under the B-CONTROL, and added to the previously accumulated scores of the previous retrievals of CONFUMAT. These scores are stored in a set of registers in the DECISION module. Now the C-CONTROL, which controls the DECISION part, takes over and the character code which corresponds to the maximum store is found and displayed. However, this will not terminate the classification process unless all the V-ELEMENTS have been processed. The processes described in this paragraph are repeated for all the remaining V-ELEMENTS and at the end the CLASSIFIER returns control to the M-CONTROL.

<u>M10: DISPLAY</u>

In the learning mode the machine will proceed to the next state, M11, on the next clock pulse. While in the recognition mode it will stay in this state as long as the user does not press the PROCEED button. The CHARACTER DISPLAYED LED is now on indicating that the

DATA OUT LEDs display the code of the recognized character. In this case the user has the choice of returning the machine to state M1 or proceeding to the next state M11. If the displayed character code is satisfactory, the user may position the switch in the DATA OUT group on the PANEL to the CORRECT position and press PROCEED returning the machine to M1, waiting for a new input. However, if the displayed code is not satisfactory, positioning the switch to the TRY AGAIN! position will make the machine proceed to M11 to 'look' at the character from a different view (vertical scanning).

3.4 The WORKING STORAGE

The WORKING STORAGE consists of the memories which hold the image of the input character and the features which are to be extracted from this image. It also contains the necessary logic for the scanning and windowing described in Chapter 2. There are three distinct memories, the IMAGEM, the NODEM and the LINKM. These memories are essentially two dimensional matrices of the same dimensions: 16x16 cells in the current implementation. A submatrix in each memory, called window, is 3x3 cells through which the contents of each memory can be processed. This window is located identically in the upper left corner of the matrices. Although this window is fixed in position, the particular organization and implementation of the memories make it possible to virtually move this window one location horizontally or vertically, in one clock cycle. Each memory consists of one or more WORKING STORAGE planes. IMAGEM and LINKM are one plane each while NODEM is four planes.

WORKING STORAGE Plane Organization:

A WORKING STORAGE plane is organized having the following design objectives in mind.

i. Resemblance, as close as possible, to the two dimensional properties of the digitized input pattern. This suggests regularity, with cell structure and accessibility in both row and column dimensions.

ii. Ease of scanning of the memories horizontally (row by row) or vertically (column by column).

iii. Ease of accessing windows. Ideally, by addressing the center cell of a window, its cells will be easily available for processing.

The universal shift register (74198) chips were found to meet the requirements stated above if they are connected in the organization shown in Figure 3.6. Each column consists of two 74198s, giving 32 chips in a 16x16 plane. In the upper left corner is the 3x3 window (W33) through which processing is performed. Inputs and outputs of an array are connected through selectors as shown in Figure 3.6. By skewing the outer row (or column) one cell position and feeding it back to the input while shifting the whole array, all the memory cells in the plane will pass a chosen fixed location. Thus, scanning horizontally (or vertically) is achieved. This fixed location is chosen to be the center cell of W33. With this technique the physically fixed window W33 can be effectively scanned over every location in the plane. To avoid end effects when the window is at the edges of the plane, the information contents of the memories are always contained

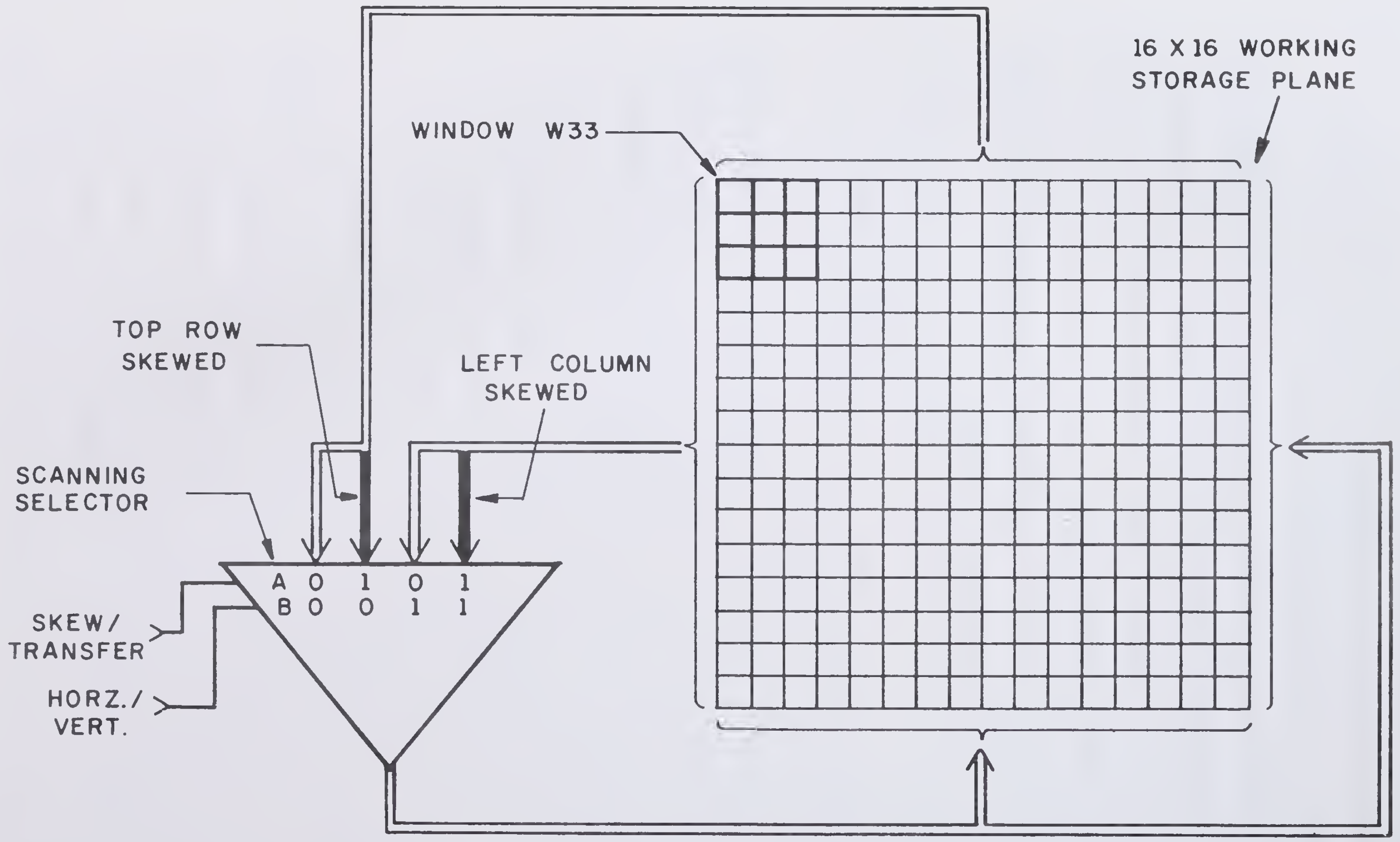


Figure 3.6 WORKING STORAGE Plane Organization

within a frame of 0's. Figure 3.7 shows examples of different scanning modes using this organization.

3.5 PRELOG Module

This is responsible for storing points in IMAGEM, preprocessing, extracting nodes and linking (Figure 3.8). The signals necessary for displaying the contents of IMAGEM memory are also generated.

The PRELOG module consists of three parts: the PREPROCESSING, NODE EXTRACTION and LINKING LOGIC part, the ADDRESSING and STORING LOGIC part and the TABLET SIMULATION and IMAGE Display Logic part (Figure 3.8). We will begin first by describing the ADDRESSING and STORING LOGIC. The two counters XI and YI, hold the coordinates of the center cell of the current window. They are connected via a selector such that they will indicate the coordinates correctly irrespective of the scanning mode, which is determined by the selecting signals HV and SKEW. For the current implementation SKEW is always held high (logic 1). XN and YN are another pair of counters which can be used to generate a cursor or loaded with the coordinates of the input point from a tablet digitizer. The cursor advancing signals are generated in the TABLET SIMULATION and IMAGE DISPLAY LOGIC. The contents of the counter pairs are compared and their coincidence signal is sent to the PREPROCESSING LOGIC. This signal indicates whether the coordinates of the point to be stored are the same as the coordinates of the center cell of the current window of IMAGEM. Upon a command from the S-CONTROL ($\overline{\text{ECORD}}$=0) the input point is inserted in the proper cell in IMAGEM.

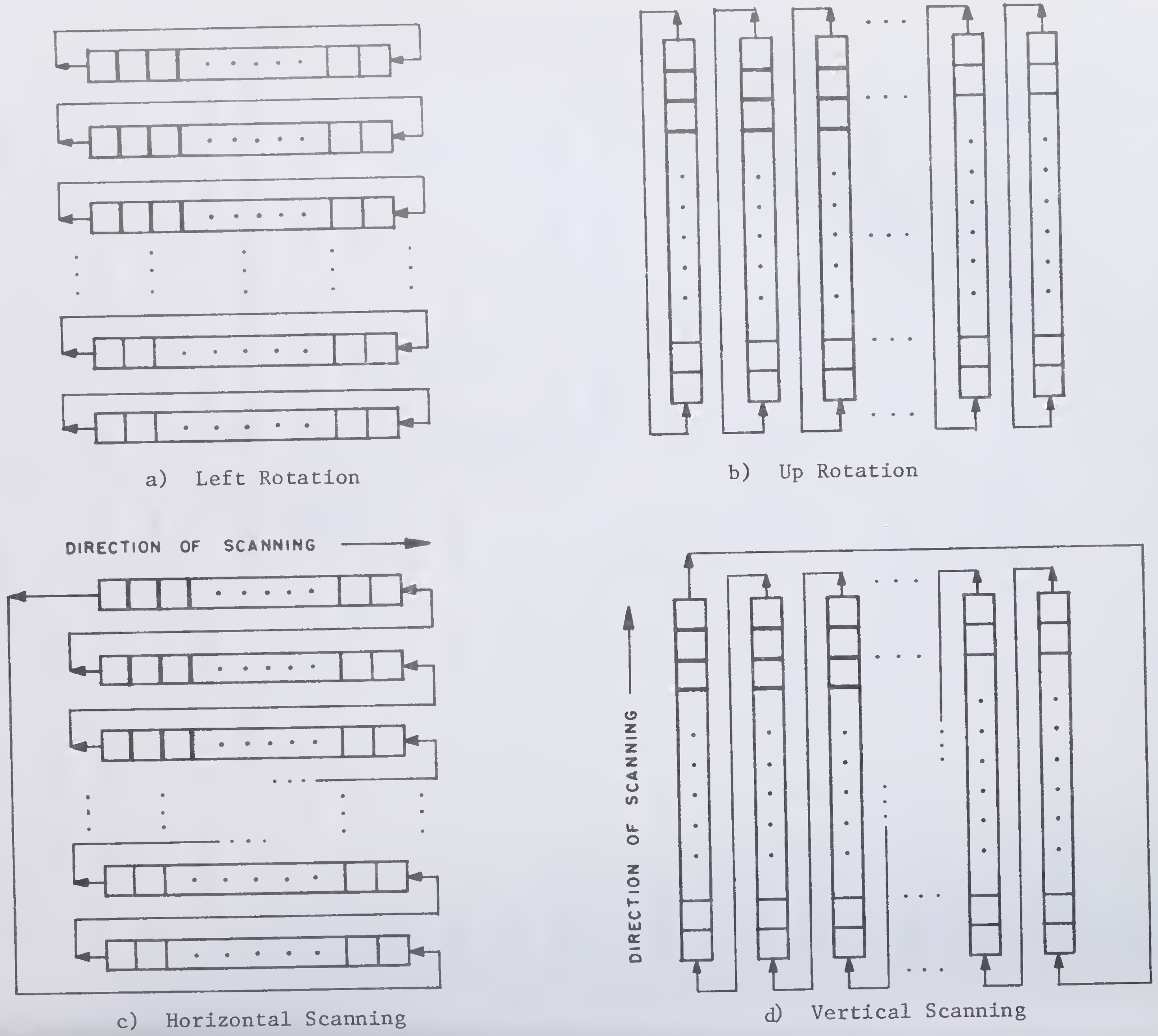


Figure 3.7 Examples of Possible WORKING STORAGE Scanning

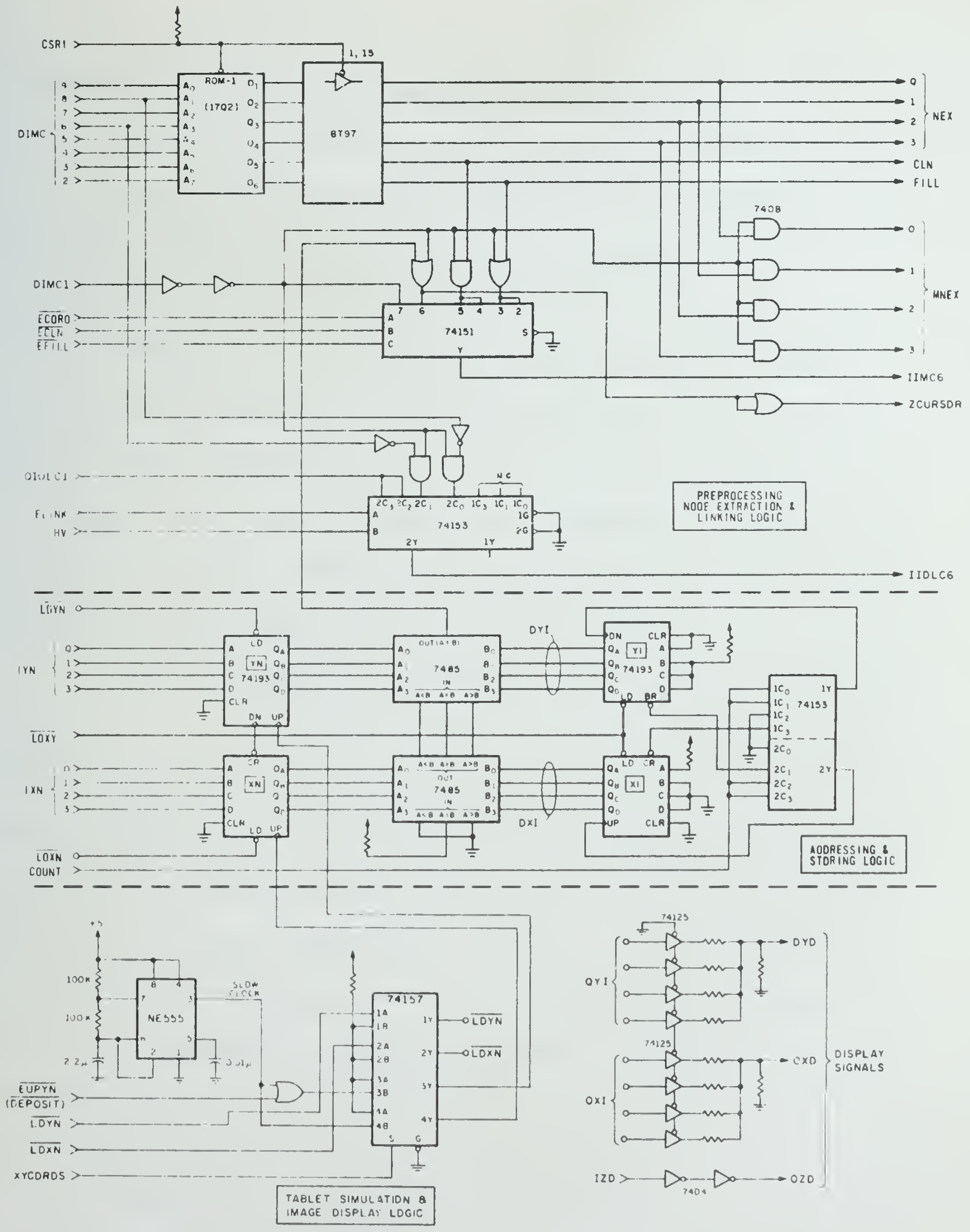


Figure 3.8 PRELOG Logic Diagram

The display signals, the counter load signals and the cursor positioning signals are generated and controlled by the TABLET SIMULATION and IMAGE DISPLAY LOGIC. When the XYCORDS signal is high the counter load signals are disabled and the cursor signals are enabled and passed to the ADDRESSING and STORING LOGIC. The XN counter is incremented by a SLOW CLOCK (.5 sec) to advance the cursor in the X-direction on the display screen. For convenience the YN counter can also be incremented by pushing the DEPOSIT button on the DATA IN group of the PANEL. When XYCORDS is low the load signals $\overline{\text{LDYN}}$ and $\overline{\text{LDXN}}$ are passed to their corresponding counters. In this case the device can accept input coordinates from a tablet digitizer under the S-CONTROL. Because an actual tablet is not available the TABLET SIMULATION XYCORD signal is kept HIGH.

Preprocessing and node extraction involve scanning the IMAGEM memory and using the outer cells of its window, OIMC2-OIMC9, to address the contents of ROM-1. ROM-1 holds the FIG, CLN and NEX-tables, as described in Chapter 2. The FILL and CLN bits of ROM-1 are used for preprocessing while the NEX bits are used for the node extraction operation. The operations of storing points in IMAGEM or preprocessing its contents can be viewed as modifying the output of the center cell, OIMC1, before storing the result in its next position in IIMC6. These modifications are simply done by a multiplexor whose output, IIMC6, is controlled by the control signals $\overline{\text{ECORD}}$, $\overline{\text{EFILL}}$ and $\overline{\text{ECLN}}$. When all these signals are inactive, high , OIMC1 is passed without change to IIMC6. The storing operation $\overline{\text{ECORD}}$=LOW, is performed by OR-ing the output of the coincidence comparator with OIMC1 before it is passed to IIMC6. Since the fill-in-gap operation is actually inserting new points, it is similarly done except that the FILL bit of ROM-1 is ORed with OIMC1 passed to IIMC6 when $\overline{\text{EFILL}}$=LOW. When $\overline{\text{ECLN}}$=LOW the clean-image operation is performed by ANDing the CLN signal with OIMC1.

To extract nodes from IMAGEM, the contents of its window outer cells, OIMC2-OIMC9, are used to address ROM-1 and the output NEX bits are ANDed with the contents of the center cell (OIMC1) to form the extracted node MNEX. MNEX is routed through the MERGER module for writing in the corresponding NODEM cell during the node extraction cycle ($\overline{\text{ENEX}}$=0).

The LINKING portion of the PRELOG module performs the linking operation by copying the contents of IMAGEM into LINKM with either the horizontal lines or the vertical lines deleted according to whether the scanning is horizontal or vertical. Two adjacent cells, including the center cell of the IMAGEM of window are tested. The next content of the center cell of LINKM window is cleared if the processed outer cell of IMAGEM window has a '1' in it. The outer cell is either IMC_6 or IMC_8, depending on horizontal or vertical scanning.

3.6 The MERGER

The MERGER will be described using Figures 3.9 and 3.10. Merging is enabled when $\overline{\text{EMERGE}}$ signal from the M-CONTROL is low. In this case the COPYING REGISTERS act as the NODEM W23 window. The logic associated with them acts as an interface between the MERGER module and NODEM by rerouting the processed cells of the window back to NODEM.

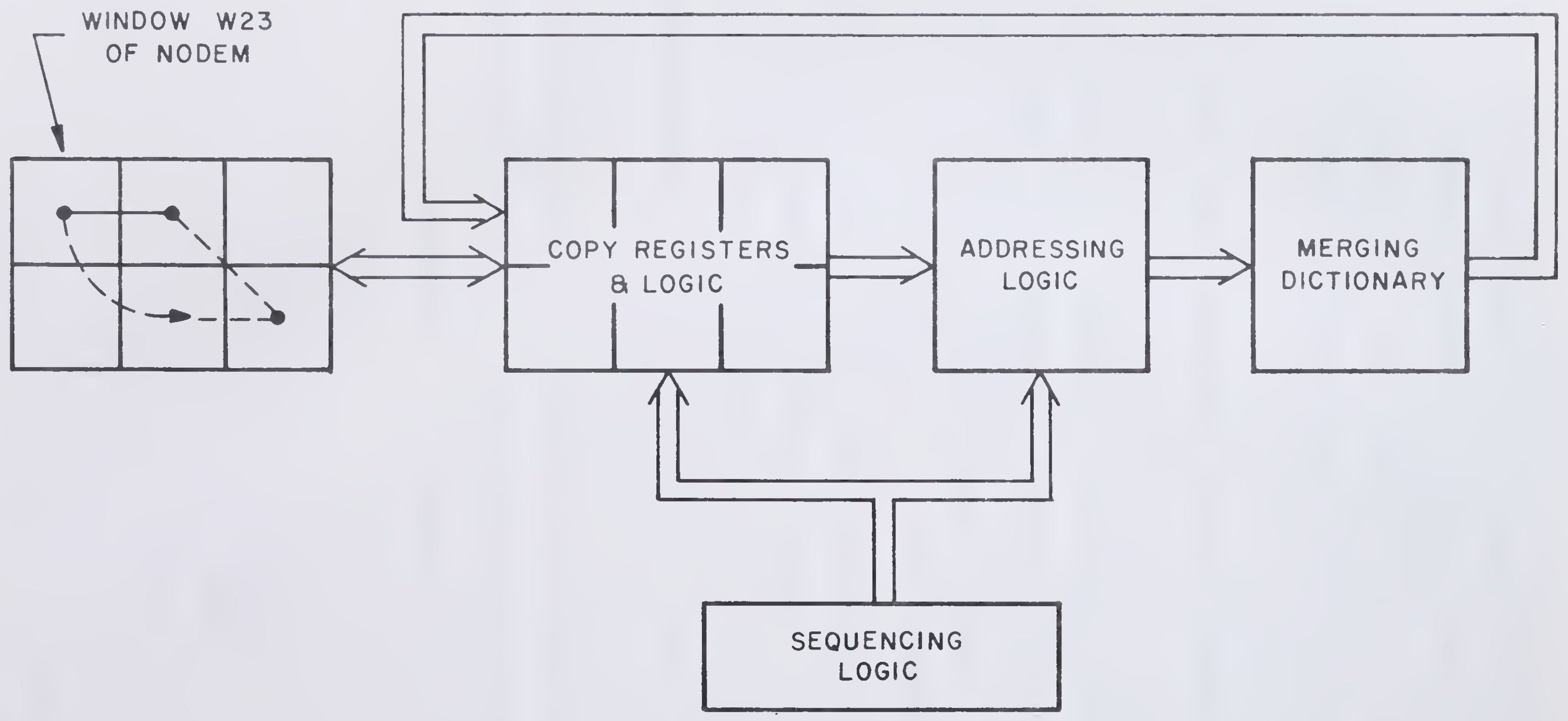


Figure 3.9 MERGER Block Diagram

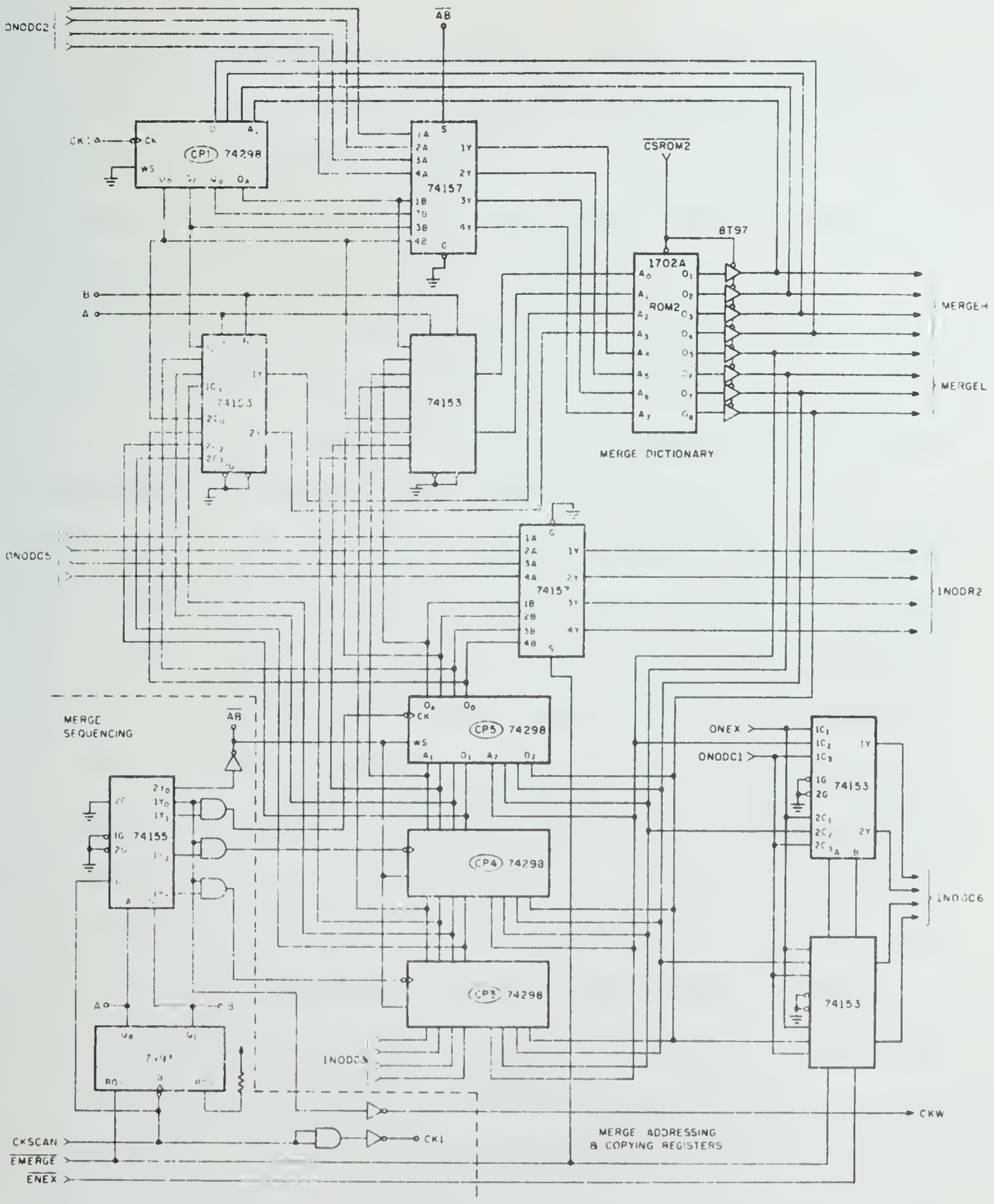


Figure 3.10 MERGER Logic Diagram

To perform merging the SEQUENCER simulates the ACSGANW12 scheme by sequencing the register clocks and selecting two cells of the copy registers to address the MERGING DICTIONARY (ROM-2) through the ADDRESSING LOGIC. Each ROM word corresponds to a merging rewriting rule for the two-cells combination which addresses it. The addressed word is used to rewrite the contents of the two cells in their new positions. In other words, one clock cycle is needed to address the ROM while the previous two cells are rewritten.

There are 4 two-cell combinations processed for each position of W23. Instead of using a clock period exclusively for positioning W23, positioning the window as well as processing the first two-cells combination are performed simultaneously. This saves a clock period: instead of five clock periods to process W23 only four are required.

The scanning clock (CKW) of the WORKING STORAGE is controlled in this module. During merging ($\overline{\text{EMERGE}}$=0) a divide-by-4 counter is enabled and the main clock is scaled down by a factor of 4 to produce CKW. However, in any other state, the counter is always reset to 0 and CKW is the same as the main clock. At the same time the COPYING REGISTERS are bypassed. In the NODE EXTRACT state ($\overline{\text{ENEX}}$=0), the MNEX output of PRELOG is routed through the MERGER to be stored in NODEM, while the merging operation is disabled.

3.7 <u>FEATURE VECTOR FORMATION SUBPROCESSOR</u>:

By the time the control is transferred to this processor ($\overline{\text{EFETV}}$=0) the WORKING STORAGE memories, IMAGEM, NODEM and LINKM will contain the different representations of the character: the preprocessed image, the local features and the horizontal or the vertical links.

The SUBPROCESSOR performs segmenting and subsegmenting the character representations of the WORKING STORAGE, as well as coding the resultant segments into the three feature subvectors ILINKV, OLINKV and TABV. These subvectors are formed concurrently and stored in their corresponding memories.

The SUBPROCESSOR consists of three main modules: the TEMPORARY SEGMENT, the FEATURE SUBVECTORS MEMORIES and the F-CONTROL (figure 3.11). The TEMPORARY SEGMENT module acts as the front end processor while the FEATURE SUBVECTORS MEMORIES hold the processed subvectors. The F-CONTROL modules directs the different operations performed by the SUBPROCESSOR. The data manipulation logic of this subprocessor is distributed among its different modules, so as to reduce the number of modules and the total number of components and interconnects.

At the same time they are scanned the WORKING STORAGE memories are segmented and each segment is compacted into three temporary vectors by the UPDATING LOGIC and stored in their corresponding registers in the TEMPORARY SEGMENT module. The contents of these vectors as well as the output of LINKM memory are transferred through the F-CONTROL module to the FEATURE SUBVECTORS MEMORIES module where they are encoded using the ENCODING LOGIC. When the end of the currently scanned character segment is encountered the encoded subvectors elements are stored into their respective locations in the FEATURE SUBVECTORS MEMORIES.

3.8 The TEMPORARY SEGMENT Module

The segmentation of the WORKING STORAGE is determined by the distribution of nodes in NODEM (Figure 3.12). The history of the segmenting flip-flop (SEGF) in the F-CONTROL module represents this distribution.

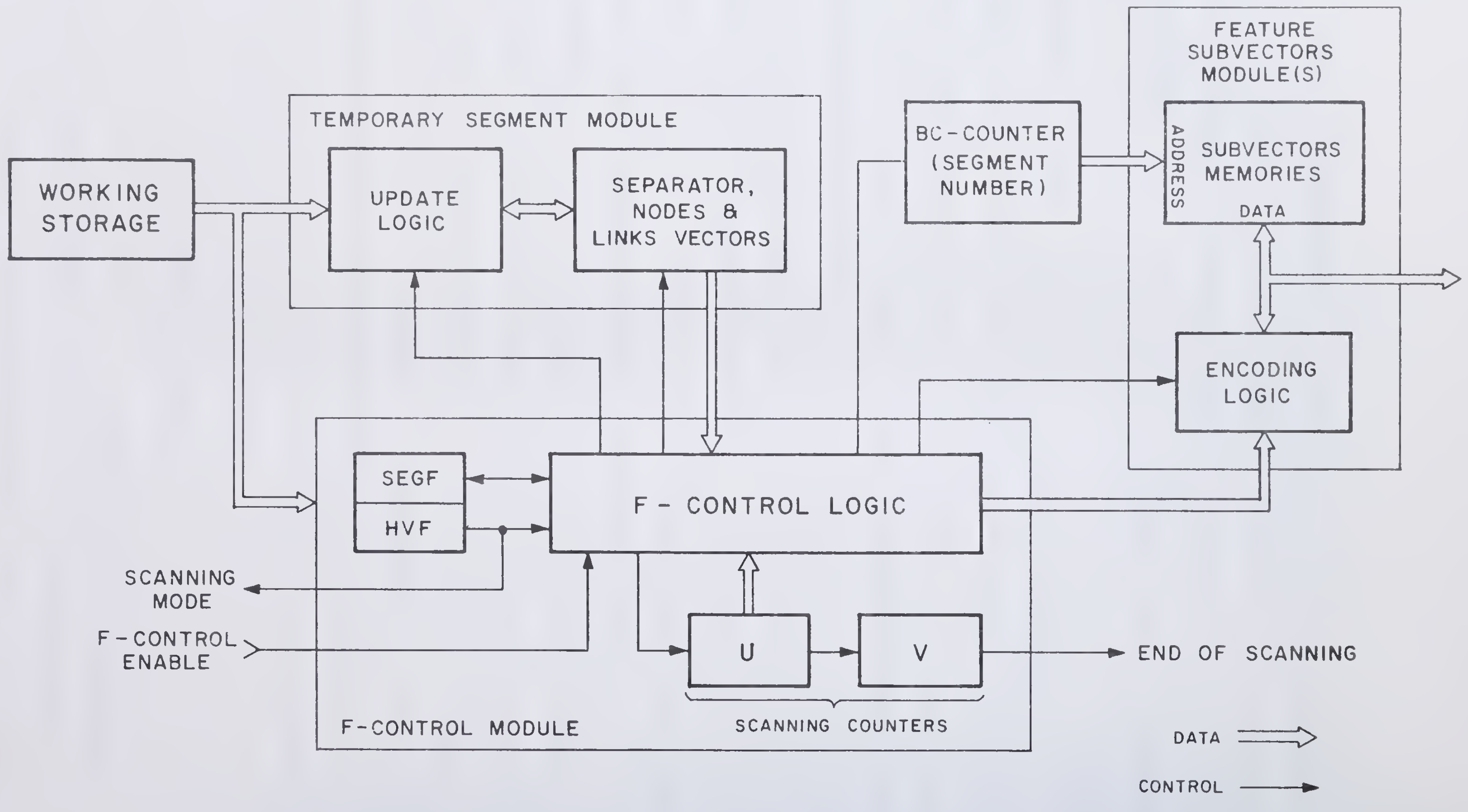


Figure 3.11 FEATURE VECTOR SUBPROCESSOR Block Diagram

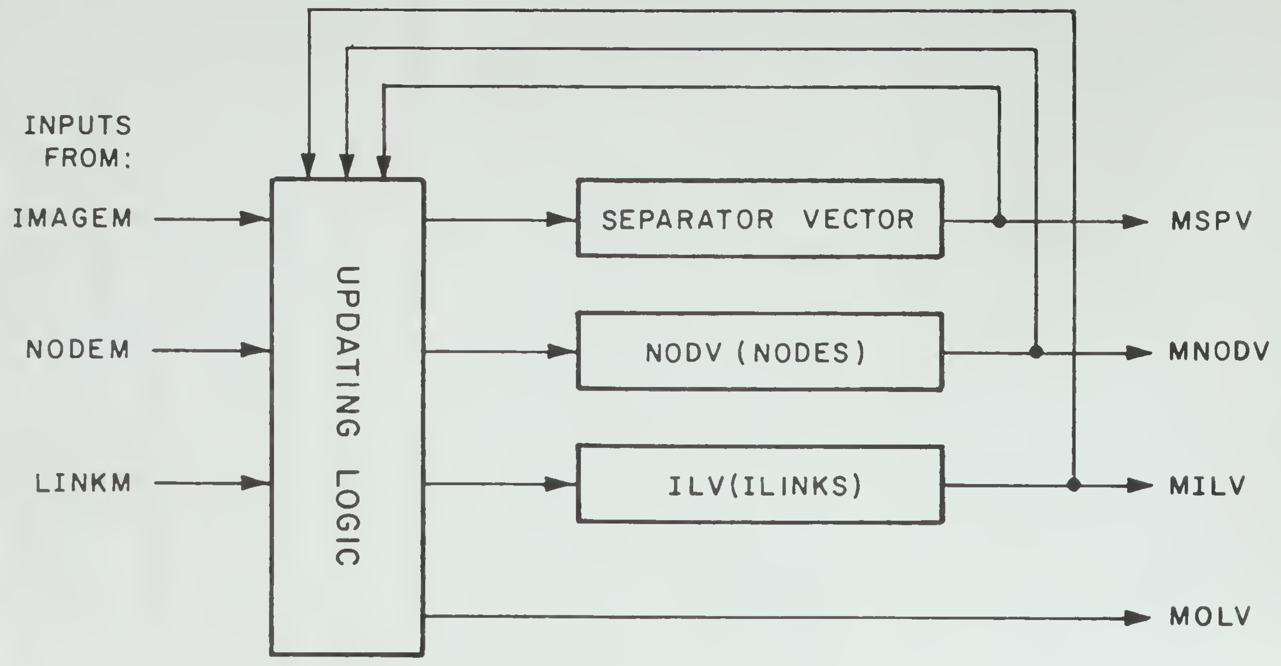


Figure 3.12 TEMPORARY SEGMENT Module Logic Diagram

According to this history the boundaries of segments are determined. Once the beginning of a new segment is found its information content is shifted into a compressed form in three vectors: the SEPARATOR vector, the ILINKS vector (ILV) and the NODES vector (NODV) (figure 3.12). The SEPARATOR vector and the NODES vector hold the information found in the IMAGEM segment and the NODEM segment, respectively. The ILINKS vector holds the first row (or column) of the currently scanned segment of LINKM.

The UPDATING LOGIC is responsible for compressing the currently scanned segment of IMAGEM and NODEM in the SEPARATOR vector and NODES vector, respectively. When it is activated by the F-CONTROL, the previous vector contents are ORed with the currently scanned row (or column) of its WORKING STORAGE segment. If the vectors are initially cleared before processing the first row (or column) of the segment, the final contents of the vectors at the end of the segment scanning will be the OR function of the rows (or columns) of the segment. The F-CONTROL will use the SEPARATOR vector contents to subdivide the scanned segment into subsegments. The feature contents of each sub-segment are extracted from the TEMPORARY SEGMENT vectors as well as the last row (or column) of the LINKM segment. This row (or column) provides the OLINKs of the character segments, which is encoded using the ENCODING LOGIC into OLINKV part of this segment. The ILINKS vector is used to generate the ILINKS of the character which is also encoded to be the ILINKV part. The NODES vector is the source of information for the TABV and NODV parts. TABV is essentially the number of nodes (or local features) in each subsegment of a segment encoded in a compacted form. NODV is the compacted local feature content of the segment. In the current implementation of this sub-processor only the ILINKV, OLINKV and TABV are generated.

Each vector is implemented as one or more variable length shift registers. The length of these vectors is programmed by a 6 position switch to be equal to the dimension of the WORKING STORAGE. A variable length implementation is chosen so that different dimensions of WORKING STORAGE can be processed by the same module. In the current implementation the switch is set to provide vectors of 16 cells each. Each cell is the same bit width as the corresponding working storage memory cell. Therefore, the SEPARATOR vector and the LINKS vector are implemented with one shift register each while the NODES vector is implemented with four shift registers.

3.9 The ENCODING LOGIC

This consists of three identical submodules for ILINKV, OLINKV and TABV subvectors. The idea is to encode the number of events in each subsegment of the ILV, OLV or NODV segment under consideration. Beginning with the first position in the 4-bit code (left) the 1's will indicate the ends of subsegments in the segment. The number of positions between the ends (two 1's) of two successive segments plus one will indicate the number of elements in a subsegment which terminates with the right hand '1' of these two 1's.

The ENCODING LOGIC consists mainly of the event COUNTER, the ENCODER and OR network feeding the INSERT REGISTER (figure 3.13). At the beginning of the segment the INSERT REGISTER is cleared and the COUNTER is reset to contatin '0'. The COUNTER and the ENCODER (selector) stimulates a

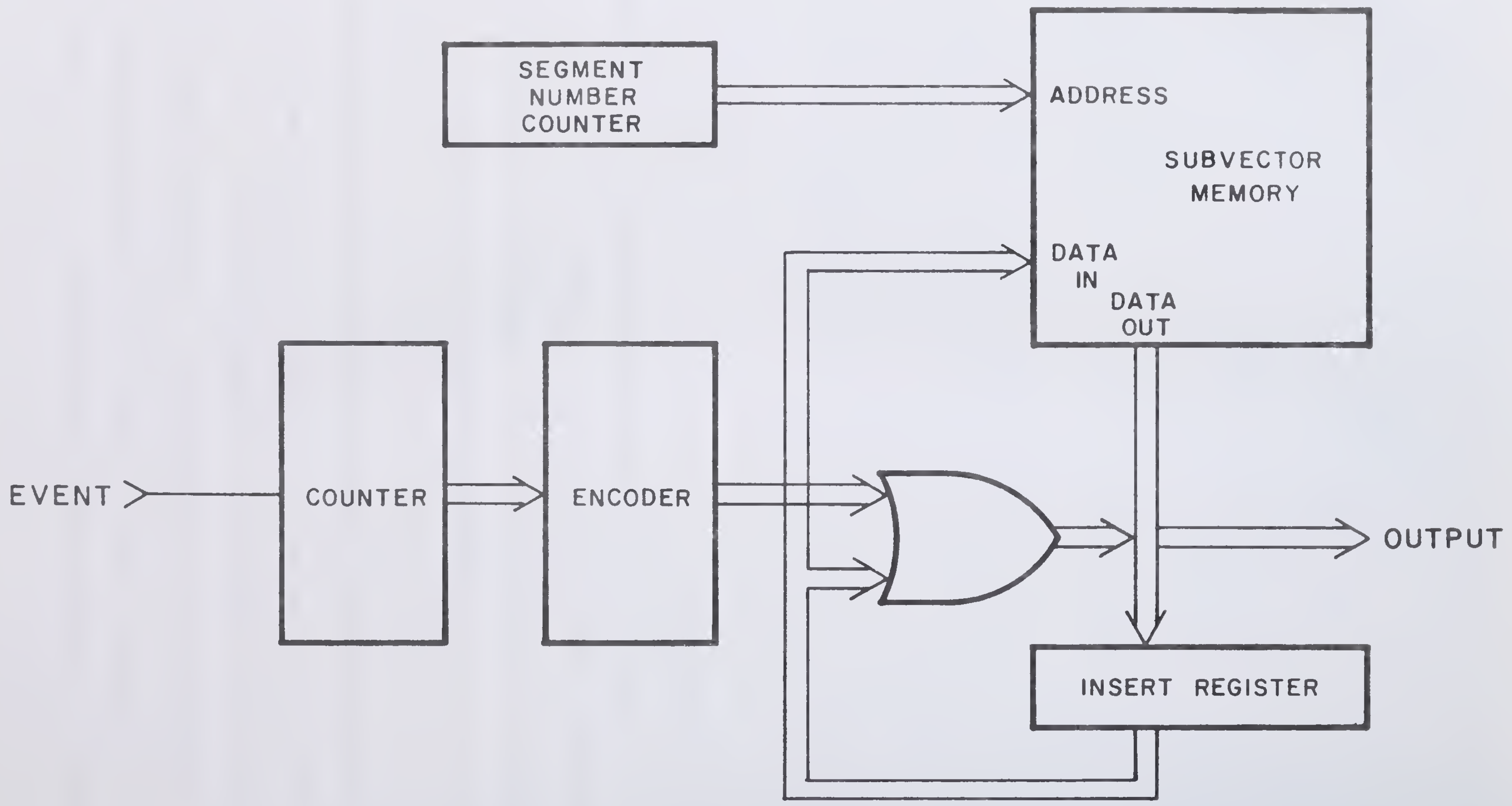


Figure 3.13 ENCODING LOGIC Block Diagram

pointer. The pointer points initially to the position immediately left of the 4-bit INSERT REGISTER. Every time an event is encountered in the scanned segment the COUNTER is incremented, thus moving the pointer one position to the right.

At the end of each subsegment, a string of 1's in the SEPARATOR VECTOR, the position of the pointer, which now equals the number of events, is OR-ed with the current contents of the INSERT REGISTER. When the scan of the current segment is completed, the different INSERT REGISTERS are written in their corresponding SUBVECTOR MEMORY locations addressed by the SEGMENT NUMBER COUNTER (BC counter).

3.10 <u>The FORMATION SUBPROCESSOR Operation</u>

The subprocessor operations (Figure 3.14) is controlled mainly by the F-CONTROL module. Besides the control logic there are two flag flip-flops (SEGF and HVF) and two loop counters (U and V). The SEGF is set to '1' whenever the next row (or column) of NODEM contains at least one node. It is always cleared at the beginning of each row (or column) (U=0). Its output is always sampled by the F-CONTROL at the end of the currently scanned row (or column). The HVF is the scanning mode flip-flop. It is set to 1 or to 0 according as the mode is horizontal or vertical.

The scanning counters U and V indicate the position of the currently processed cell. The U-counter indicates the column or row while the V-counter indicates the row or column position for the cases of horizontal or vertical scanning modes. The U-counter is incremented by the WORKING STORAGE scanning clock (CKW). The carry output of the U-counter indicates the end of the currently scanned row or column

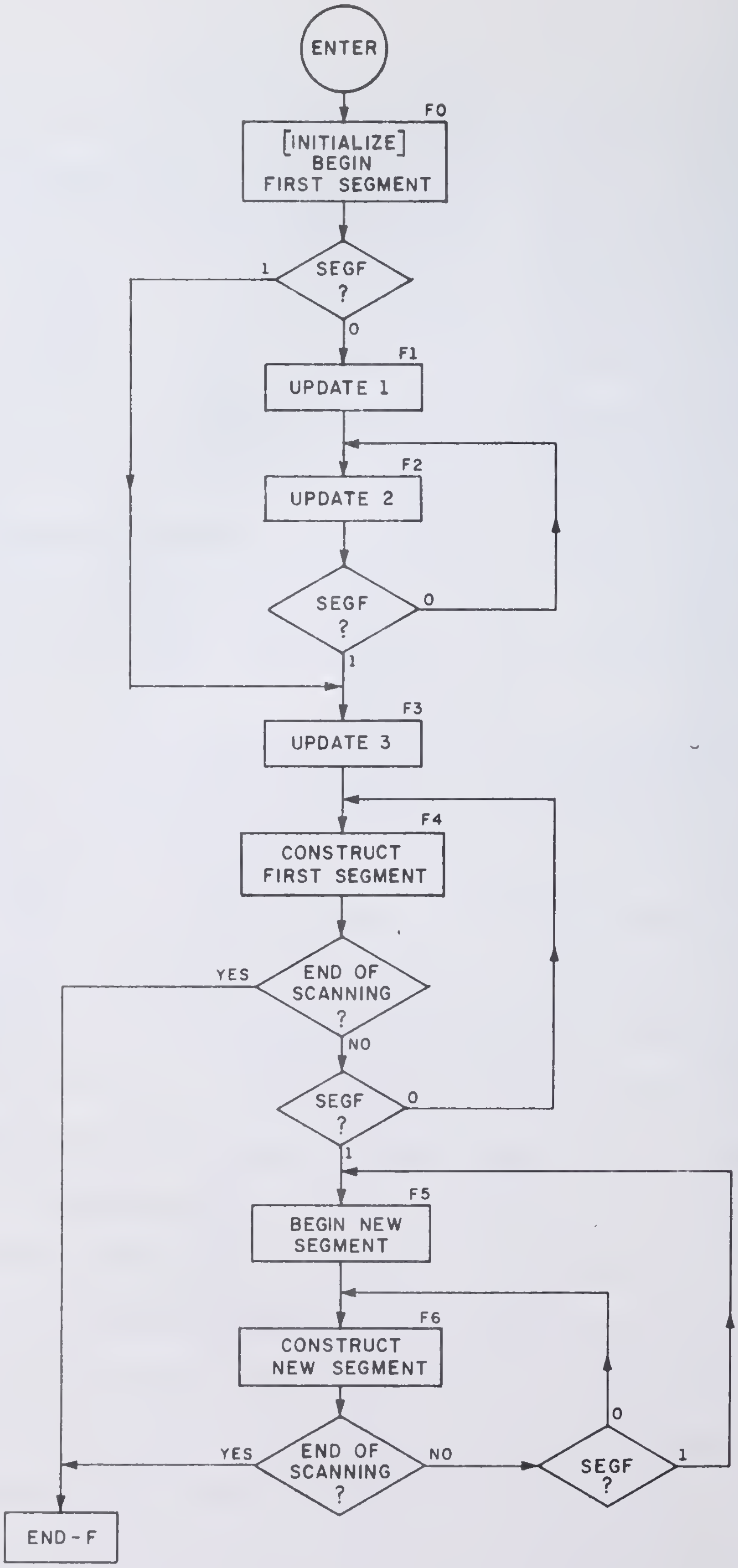


Figure 3.14 F-CONTROL Flow Chart

and it also increments the contents of the V-counter. The carry of the V-COUNTER ($\overline{\text{CRV}}$=0) signals the end of scanning.

Before describing the operation of the F-control we observe the following. First, the memory buffers and counters of the ENCODING LOGIC, in the FEATURE VECTOR module, are always cleared at the falling edge of the master clock when U=0. Second, the BC counter holds the number of the currently scanned segment. It is updated by an incrementing pulse from the F-CONTROL whenever a new segment is encountered. At the end of scanning it contains the total number of segments minus one. While scanning the WORKING STORAGE the following operations are performed in the subprocessor.

FO: INITIALIZE (BEGIN FIRST SEGMENT):

The TEMPORARY SEGMENT vectors SEPV, NODV and ILV and the BC counter are cleared. The control remains in this state for one clock period. At the end of state the output of SEGF is sampled. Since SEGF is not cleared yet, its contents will indicate whether the row (or column), which is about to be processed, has a node in it or not. If SEGF=1 the control goes to state F3. Otherwise, it will proceed to state F1.

F1: UPDATE1:

The processing of the first segment (BC=0) begins in this state. ILV will remain cleared. This indicates that there is no connection between this segment and the segment before it; because it is the first segment. The updating of SEPV and NODEV begins here. The duration of this state is one clock period.

F2: UPDATE2:

The procedure which has begun in F1 is continued here. This means that SEPV and NODV are updated. Since SEGF is not tested at the end of F1, it means that the current row (or column) is ORed in the SEPV and NODEV whether it contains a node or not. At the end of F2 ($\overline{\text{CRU}}$=0), SEGF is sampled. If SEGF=0 the control remains in F2, otherwise it proceeds to F3.

F3: UPDATE3:

This state indicates the first occurrence of a row (or column) which has a node in it. Updating is also continued here but only for one row (or column).

F4: CONSTRUCT FIRST SEGMENT:

The updating of the SEPARATOR and NODES vectors is continued. At the same time the encoding of the different subvector elements is performed. At the end of this state the constructed elements in the memory buffers are written in the memory locations addressed by BC. This means that they will reside in location '0'. The control will remain in this state unless SEGF is 1. If there are no other segments the scan will end by branching to F7. The subsequent segments are processed during states F5 and F6.

F5: BEGIN NEW SEGMENT:

The processing of a new segment, other than the first, begins here. To indicate this fact BC is incremented. The TEMPORARY SEGMENT vectors are cleared at the beginning of the first row (or column) before storing the new information in them. The remainder of the state is devoted to updating the above vectors. The ILV will contain the first row of the new segment of LINKM. To insure continuity and consistency the last row (or column) of the last segment is ORed with the now new row (or column) of the new segment. The NODES vector is updated as usual.

F6: CONSTRUCT NEW SEGMENT:

The operations performed in this state are similar to that performed in F4 except they are performed for new values of BC which is not zero anymore.

F7: ENDF:

When the end of scan is reached ($\overline{\text{CRV}}$=0) the control is transferred to this state. On the next clock cycle the control is transferred to F0 waiting for the next activation of the F-Control ($\overline{\text{EFETV}}$=0).

3.11 The CLASSIFIER

Besides the controls the CLASSIFIER consists mainly of three parts; the V-ELEMENT CONSTRUCTION part, the DATA STRUCTURING part, and the DECISION part (Figure 3.15). The first part reads the FEATURE VECTOR memories, constructs a V-ELEMENT and sends it to the DATA STRUCTURING part for further processing. The DATA STRUCTURING part accepts this V-ELEMENT and tries to find a match for it among the stored V-elements in its store. There are two modes of operation for this module: the learning mode, and the recognition mode. If the matching is not successful and the module is in the learning mode the new V-ELEMENT is stored with its character code in a new location in the FEATURE POINTER LIST memory (FEPOLIST).

If the matching is successful and a character code is stored in the same word of FEPOLIST two possibilities arise. The first

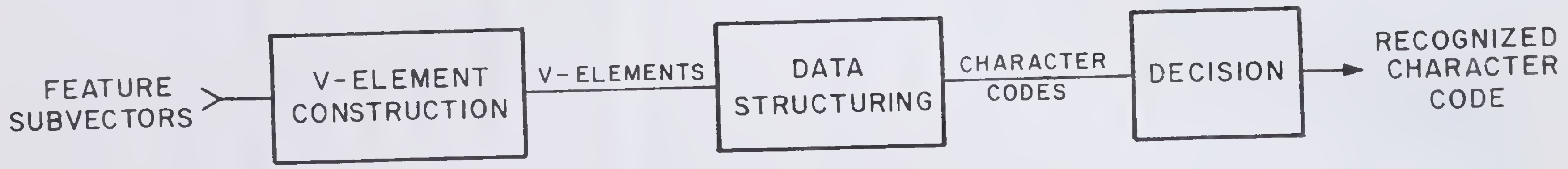


Figure 3.15 INCOM CLASSIFIER Block Diagram

possibility is that the stored character is the same as the new input character code. The next V-ELEMENT is produced and the above processes are repeated. The second possibility is that the input character code is different from the stored character code. In this case, these two codes are decoded into one CONFUMAT word. The address of this word will be stored as a pointer in the corresponding FEPOLIST word. A last possibility arises when the V-ELEMENT matches the feature part in a word in FEPOLIST but the POINTER part addresses a CONFUMAT word. In this case, the new character code is encoded and added to the CONFUMAT word by setting to '1' the corresponding character bit.

In the recognition mode, each new V-ELEMENT is matched against all the V-ELEMENTS stored in FEPOLIST. If the matching is successful and a character code is stored in the matched FEPOLIST word, this code will be displayed as the recognized character code. If a pointer is stored instead, the addressed word of CONFUMAT will be read and its contents are sent to the DECISION module. When all the V-ELEMENTS are processed the decision logic will display the character which possesses the maximum matching score.

Four control modules supervise the different operations performed by the CLASSIFIER. These modules are the A-control, the B-control, the V-control and the C-control. The A-control acts as the master control for the remaining modules as well as supervising the operations of the DATA STRUCTURING part. The B-control complements the operations of the A-control. The V-control supervises the operations of the V-ELEMENTS CONSTRUCTION module. Physically the

A-control and the B-control are on two separate cards while the V-control and C-control occupy a third card.

In the following sections, the data manipuator and memory modules of the CLASSIFIER will be described separately. The constructs of each module are described and their functions are discussed briefly. This should prepare the reader for the detailed discussion of the CLASSIFIER operations. These operations are distributed among the control modules. The operations of the satellite controls (V-CONTROL, C-CONTROL and B-CONTROL) will be described first. The A-CONTROL which acts as the master control for the CLASSIFIER is described last. This will sum up the description of the CLASSIFIER.

3.12 The V-ELEMENT CONSTRUCTION Module

This consists of three parts; the VECTOR ADDRESSING part, the V-SCHEMES MEMORY part and the V-MULTIPLEXORS (Figure 3.16). The V-SCHEMES MEMORY (VMEM) is a writable memory (16 bytes) in which the construction schemes are stored. Each byte, when read in the memory buffer (BV), controls the outputs of the VECTOR ADDRESSING and the V-MULTIPLEXORS parts. The contents of either the TC counter ot the BC counter will address the FEATURE SUBVECTOR memories outputs, some of which are selected to be passed through V-MULTIPLEXORS to R-BUS.

A V-ELEMENT constructed by this module consists of three bytes. The first byte is the identification byte (ID); it names the construction scheme used to construct the remaining two bytes, the scanning mode and a segment location tag (contents of BC or TC). The other two bytes constitute the feature part, which usually consists of a combination

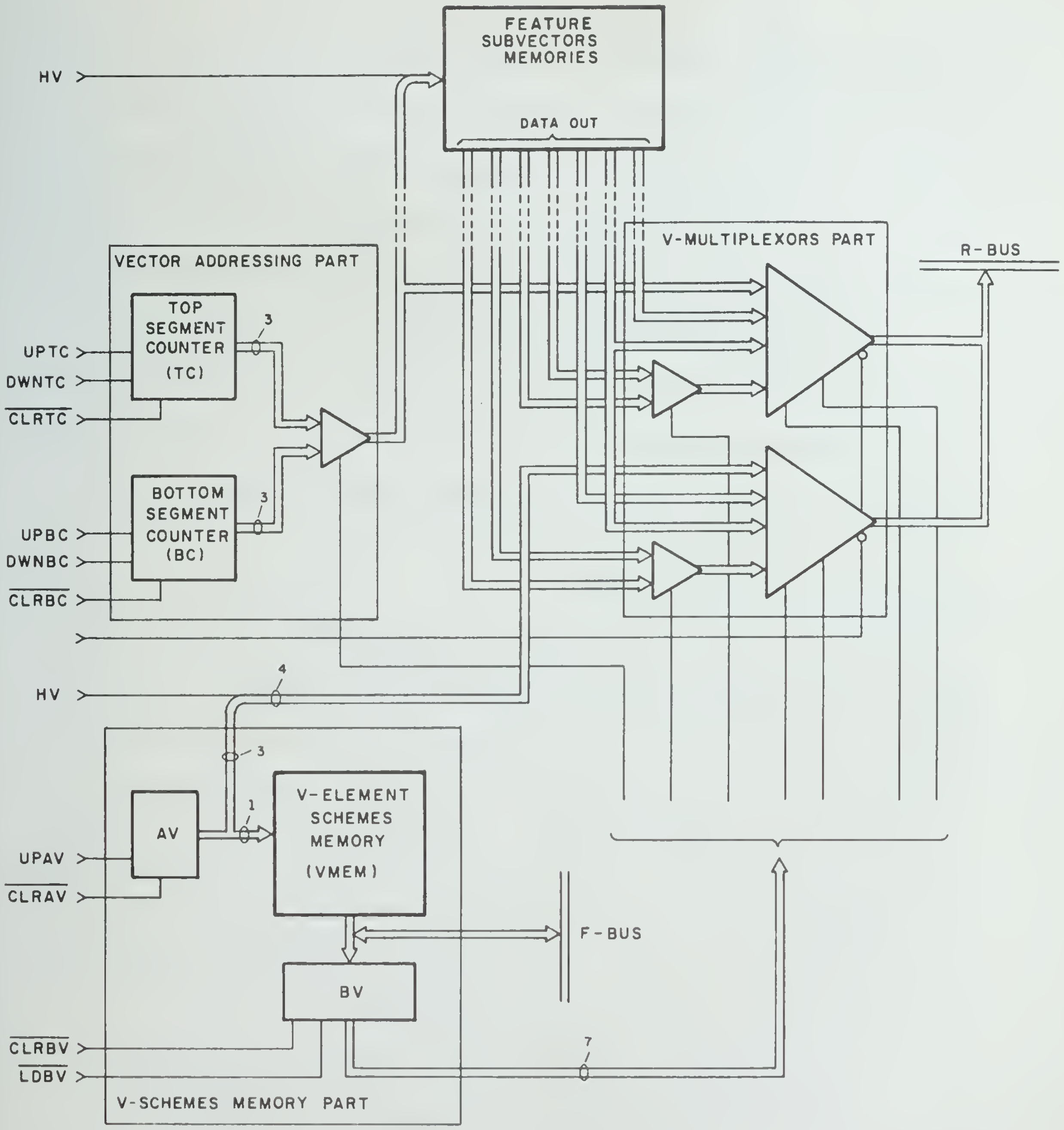

FEATURE SUBVECTORS MEMORIES
DATA OUT
HV
VECTOR ADDRESSING PART
TOP SEGMENT COUNTER (TC)
UPTC
DWNTC
$\overline{\text{CLRTC}}$
3
BOTTOM SEGMENT COUNTER (BC)
UPBC
DWNBC
$\overline{\text{CLRBC}}$
3
V-MULTIPLEXORS PART
R-BUS
HV
4
3
1
AV
UPAV
$\overline{\text{CLRAV}}$
V-ELEMENT SCHEMES MEMORY (VMEM)
F-BUS
BV
$\overline{\text{CLRBV}}$
$\overline{\text{LDBV}}$
7
V-SCHEMES MEMORY PART

Figure 3.16 V-ELEMENT CONSTRUCTION Module Block Diagram

of several elements from different feature subvectors. The feature part will be produced byte by byte using two bytes (one byte each) from VMEM. These construction schemes bytes are addressed by the AV register/counter. The three most significant bits of AV are considered the name of the construction scheme and are connected to the input of the V-MULTIPLEXORS. TC and BC counters will contain at most the maximum number of segments--less than 8. Therefore, the identification byte will contain 3 bits of AV, 3 bits of TC/BC, 1 bit of HV and a bit reserved for storing the type of pointer in FEPOLIST memory.

3.13 The DATA STRUCTURING Part

This consists mainly of the FEATURE POINTER LIST (FEPOLIST) memory and the CONFUSION MATRIX (CONFUMAT) memory modules (Figure 3.17). FEPOLIST is a RAM of 256 4 byte wide words; three bytes contain the F-part and the fourth byte contains the P-part. CONFUMAT memory is another RAM of 256 words. In the current implementation, each word has a maximum width of 4 bytes; thus 32 characters can be processed. However, a dip switch on the DECISION module can change the word width from one byte to 4 bytes in one byte increments.

Two up-down counters (MF and MC for FEPOLIST and CONFUMAT, respectively) act as memory address registers. LF and LC are two registers which act as top-of-stack pointers to FEPOLIST and CONFUMAT, respectively. When each of their memories is not full, they point to the next available memory location. LF is also useful when searching FEPOLIST by loading its contents into MF and using this as a loop counter.

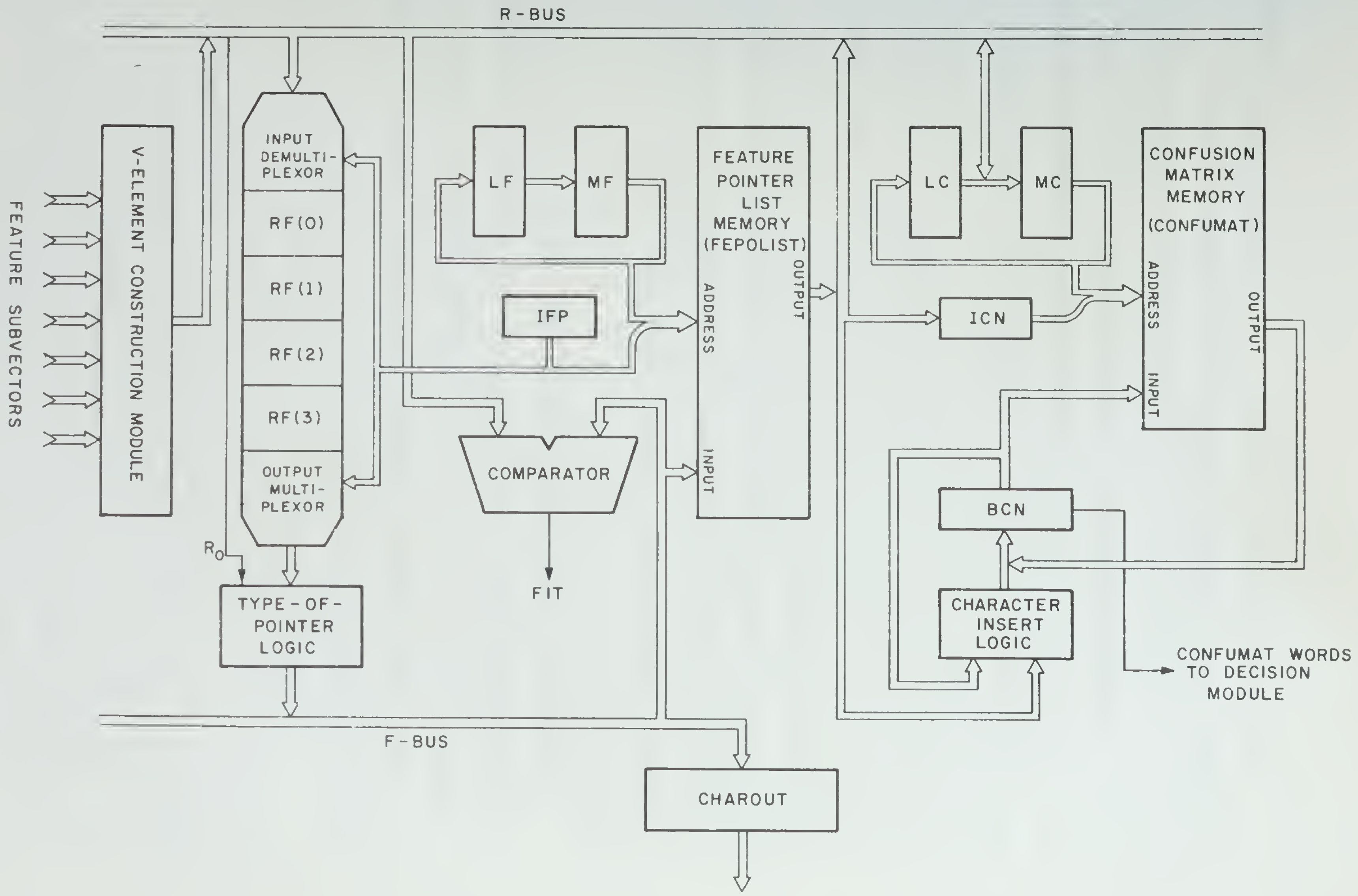


Figure 3.17 DATA STRUCTURING Part Data Flow Diagram

IFP and ICN are index counters which address bytes in FEPOLIST and CONFUMAT words, respectively. While IFP addresses a byte in FEPOLIST it also addresses a corresponding register in the register file RF. Each register acts as a memory buffer for a FEPOLIST byte. RF(0), RF(1) and RF(2) hold a V-ELEMENT from the V-ELEMENT CONSTRUCTION part or an F-part from FEPOLIST. RF(3) may hold an input character code or a P-part.

The first bit of the first byte of the F-part is the type of pointer bit (TP). The type of pointer flip-flop (TPF) always holds the TP bit of the currently accessed FEPOLIST word. To write a new TP the contents of TPF is modified first. Then, while writing RF(0) in FEPOLIST its bit-0 is overridden by TPF. In all other occasions the outputs of RF(1), RF(2) and RF(3) are passed to F-BUS without modifications.

BCN is the memory buffer register of CONFUMAT. With the aid of CHARACTER INSERT LOGIC, it also plays an important role in encoding a character code into its equivalent bit position before storing it in CONFUMAT. The contents of BCN can also be shifted serially to the DECISION module during recognition.

To encode a character, its code is presented on R-BUS. Assuming the code is 5 bits wide, the two most significant bits will be used to address a byte of an addressed CONFUMAT word. They are loaded into the index register ICN. The remaining 3 bits will address a bit in BCN. This bit is set to '1' by the CHARACTER INSERT LOGIC. To complete the encoding operation, the new contents of BCN are written in the byte addressed by ICN in a word addressed by MC.

3.14 The DECISION Module

Its main components are the INPUT CHARACTER (CHAR) register/counter, the SCORE ACCUMULATOR (SAC), the SCORE ADDER, the MAXIMUM SCORE REGISTER (MSR) and SCORE COMPARATOR (Figure 3.18). CHAR acts as a register for the input character code or a loop counter for the C-CONTROL, for the case of learning or recognition, respectively. In the second case the initial contents are loaded from a switch which is set to contain the CONFUMAT word width. The same switch controls the length of SAC.

The SCORE ACCUMULATOR (SAC) holds the accumulated scores resulted from adding all the CONFUMAT words read during a recognition cycle. This is done, by recirculating SAC and adding a new CONFUMAT word to the previous contents of SAC using the SCORE ADDER.

To decide which is the recognized character code, SAC is recirculated while inhibiting the ADDER. Under the C-control supervision MSR is compared with each cell of SAC. CHAR will keep track of the position of the compared SAC cell. The maximum score is always loaded in MSR while the corresponding character code (CHAR contents) is passed to the R-BUS to be sent through RF(3) to the output character register (CHAROUT) for display.

3.15 The V-CONTROL

This controls the operation of the V-ELEMENT CONSTRUCTION module. Upon receiving a request signal from the A-CONTROL a V-ELEMENT is constructed and stored in the RF file registers in the FEPOLIST module--RF(0), RF(1) and RF(2).

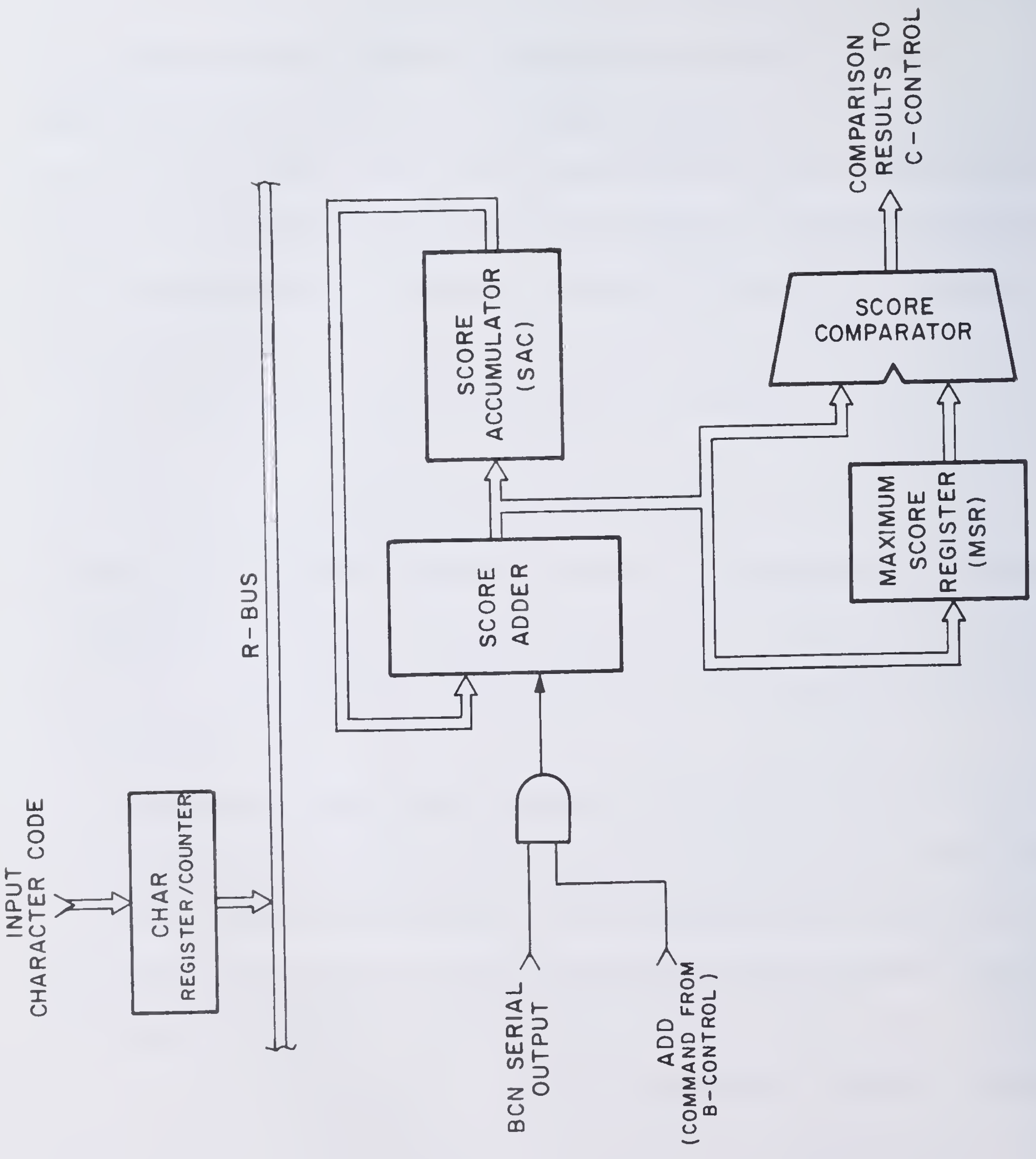


Figure 3.18 DECISION Module Block Diagram

The V-CONTROL has been designed such that the V-ELEMENTS constructed are the results of 'looking at' the character, represented by its feature subvectors, from two opposite sides; top and bottom or left and right, for horizontal or vertical scanning, respectively. This is done by scanning the subvectors using the TC and BC counters as pointers to both ends of the character. One of them is chosen at a time by the construction schemes to address two 4-bit subvector elements for constructing a V-ELEMENT byte.

At the beginning TC contains 0 while BC contains the maximum number minus one of the character segments of the current scanning scheme (horizontal or vertical). TC is incremented and BC is decremented to move these pointers from both ends of the character. For each construction scheme a V-ELEMENT is constructed for each position of these counters. To avoid duplication of V-ELEMENTS, the construction is done until TC and BC point to the middle segment of the character (BC=TC when BC is even or BC=TC+1 when BC is odd).

To try a new construction scheme (a new even value of AV), the TC and BC are reset to their initial values. V-ELEMENTS are then constructed upon request, using the new construction schemes for each position of TC and BC.

When all possible schemes have been tried (when AV overflows) the V-ELEMENT CONSTRUCTION is terminated. The A-CONTROL is notified and the CLASSIFICATION will be terminated as well.

The detailed operation of this module is described using the flow chart in Figure 3.19. For each new V-ELEMENT the VMEM buffer (BV) is cleared in order to pass the identification byte to the R-BUS. To write this byte in RF(0) the index register IFP is also cleared. We should remember that two construction schemes bytes are needed to write the feature part in RF(1) and RF(2). The first byte is addressed by the new AV (even value) while the second byte is addressed by incrementing AV (odd value). Now, the detailed description of operation follows (Figure 3.19).

VO: WAIT

The control remains in this state waiting for the enabling signal ($\overline{\text{ENV}}$) to be low. The control will also go to this state whenever a reset pulse ($\overline{\text{CLRV}}$) is received or when the END-V is reached and the classifier has completed operation ($\overline{\text{ENDCLASFY}}$=0).

V1: INITIALIZE

To prepare for constructing the identification byte the BV register is cleared making the selector lines of the MULTIPLEXORS low. The HV signals, the TC counter output and the AV address counter output are ready to be passed to the R-BUS. The IFP counter is also reset to 0 so that the identification byte will be stored in location 0 of the register file RF.

V2: WRITE R-BUS in RF

The MULTIPLEXORS are enabled in this state. The information on the R-BUS is written in RF. If IFP=0 the identification part appears on the R-BUS. Otherwise, one byte of the feature part will appear and will be stored in the RF registers RF(1) or RF(2) according to the contents in IFP. If IFP=0 the address register AV remains the same,

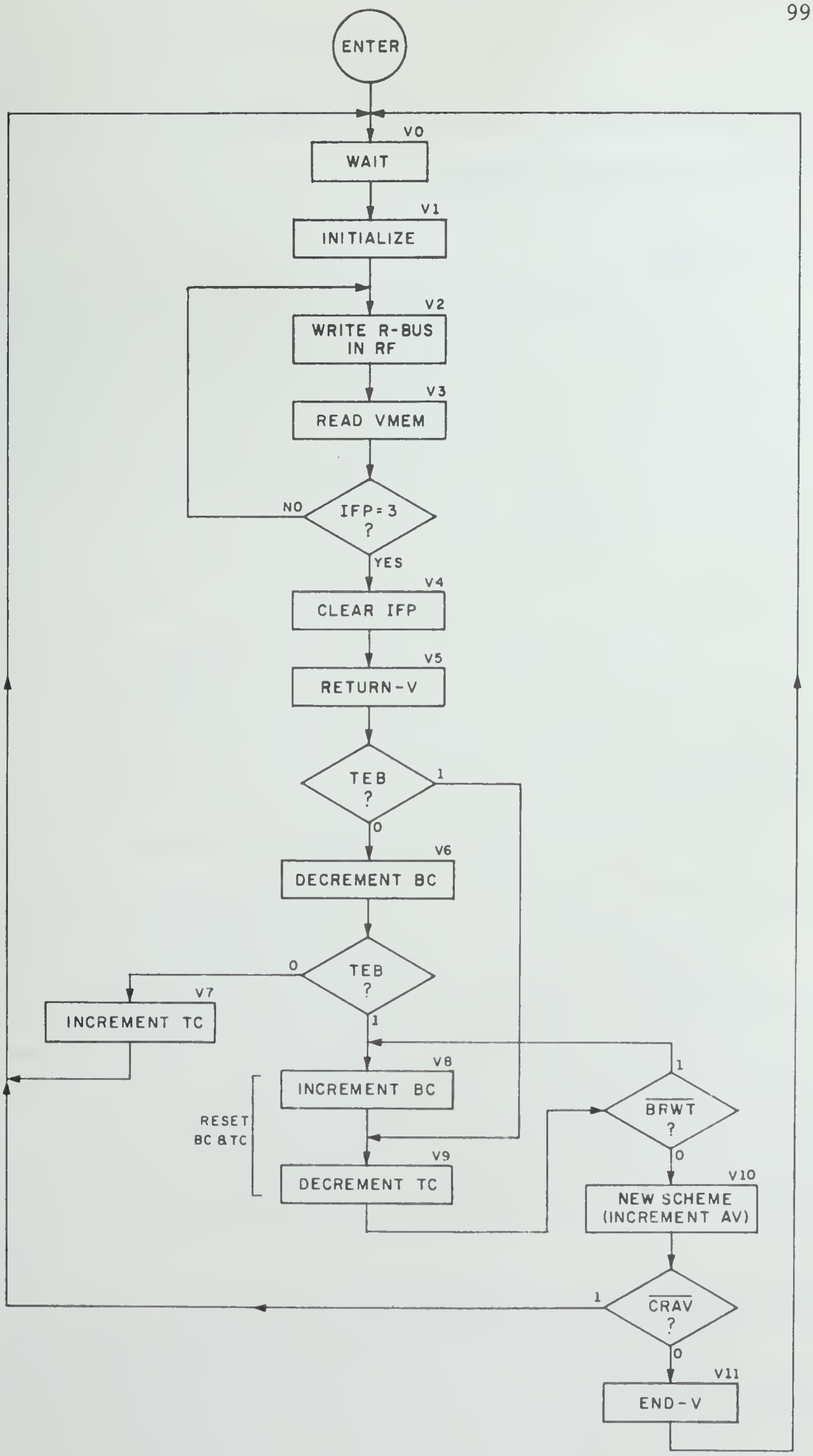


Figure 3.19 V-CONTROL Flow Chart

otherwise, it will be incremented at the end of the state to address the next byte scheme of VMEM.

<u>V3: READ VMEM</u>

The addressed byte in VMEM is loaded into BV. The IFP counter is also incremented. The completion code for constructing a V-ELEMENT is tested by sampling the outputs of IFP. After the three bytes of V-ELEMENT have been stored in RF (IFP=3) the control transfers to V4. Otherwise, it transfers to V2.

<u>V4: CLEAR IFP</u>

To begin processing the constructed V-ELEMENT when returning to the classifier A-Control, the IFP counter is cleared.

<u>V5: RETURN-V</u>

The V-control signals the A-control of completion of operation. The V-ELEMENT bytes reside in RF waiting for further operation.

At the end of V2 the condition of TEB (TC equal BC) is tested and the control is transferred either to V_6 or V_9 according as TEB is 0 or 1. If TEB is 1 this means that the current scheme addressed by the scheme number held in AV has been tried for all TC and BC combinations, allowing for TC $\leq$ BC only. In this case the control is transferred to V_0. In the case of TEB is 0 the branching is to V_6.

<u>V6: DECREMENT BC, V7: INCREMENT TC</u>

BC and TC can be considered as two pointers which point to two opposite segments of the character. They are not allowed to overlap so that the combinations will not be repeated. From this comes the condition that TC $\leq$ BC. Coming from state V_5 it has not been determined yet if TC+1=BC. This is to assure that when BC is decremented and TC is incremented the condition TC $\leq$ BC still holds. For this reason BC is decremented first with the new value of BC. If they are not equal (TEB=0) TC is incremented in V7 to get a new combination of BC and TC with the condition TC $\leq$ BC. The counter AV is also decremented to return it to the first byte of the current scheme. The V-control goes to the WAIT state V0 waiting for another request from the A-control. And, the current scheme will be used again to process the segments addressed by BC and TC.

V8, V9: RESET BC and TC

Branching occurs to one of these states in order to return BC to its original value which is 0. This prepares for a new scheme to be tried with new BC and TC combinations.

V10: NEW SCHEME (INCREMENT AV)

TC is cleared in this state because of the underflow which occurred because of V9. The AV counter is incremented to address the new scheme. However, at the end of the state the AV overflow ($\overline{\text{CRAV}}$) is tested for completion of all schemes. If all the schemes have not been tried the control branches to V0. Otherwise, it will go to V11.

V11: ENDV

The control reaches this state, when there is no further schemes to be tried on the current character in the current scanning mode.

V-control remains in this state until a completion signal from A-control (ENDCLASFY) is received. This will make V-control to branch to V0.

3.16 The C-CONTROL

The primary purpose of this module is to search the SCORING ACCUMULATOR (SAC) for the maximum score. When the maximum score is found it is stored in the MACIMUM SCORE REGISTER (MSR) and the corresponding character code is passed via the R-BUS to RF(3) and from RF(3) via the F-BUS to the CHAROUT register for display. The description of operation follows (Figure 3.20).

C0: WAIT

The C-control will remain in this state waiting for an enabling signal from the B-control to go to the INITIALIZE state C1. Resetting the C-control counter will also put the control in this state.

C1: INITIALIZE

The IF counter is set to contain 1's to address RF(3). And, the MAXIMUM SCORE REGISTER (MSR) is cleared to contain 0.

C2: LOAD MSR and LOAD RF(3)

Branching to this state means that each score in SAC is greater than the score already stored in MSR. MSR is loaded by the new maximum score from SAC and the corresponding character code, held in CHAR counter, is loaded into RF(3). The CHAR counter is decremented at the end of this state. The underflow signal ($\overline{\text{BRWCHAR}}$) is also tested. If $\overline{\text{BRWCHAR}}$=0 this means that all SAC contents have been processed and the

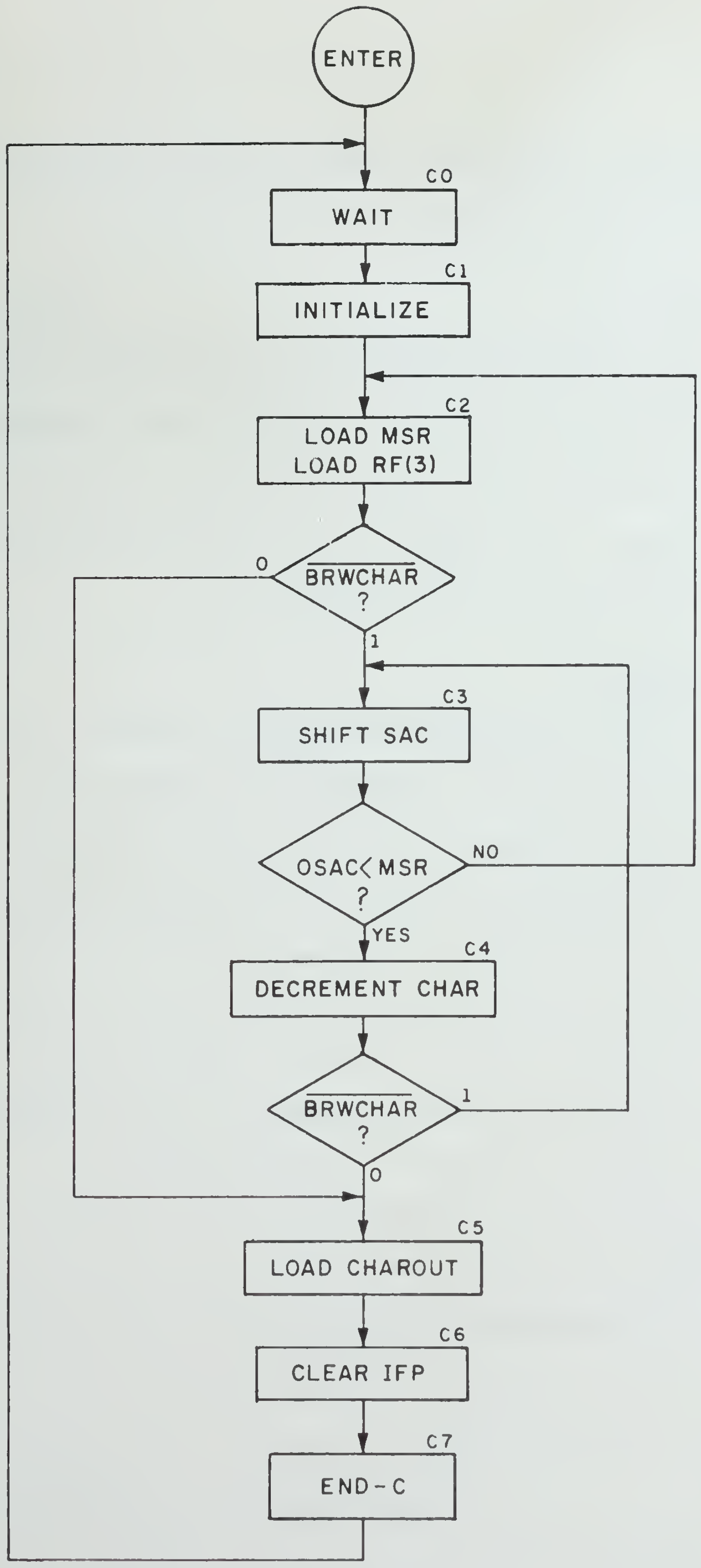


Figure 3.20 C-CONTROL Flow Chart

control jumps to C_5. Otherwise, there are some locations in SAC to be tested and the control proceeds to the next state C_3.

C3: SHIFT SAC

SAC is clocked to shift the next unprocessed location to appear on its output. This output is tested against the current contents of MSR. If SAC output is greater than or equal to the contents of MSR, MSR and RF(3) have to be updated. In this case the control is returned to C2. Otherwise, the control proceeds to C4.

C4: DECREMENT CHAR

The CHAR counter is decremented at the end of this state and the $\overline{\text{BRWCHAR}}$ is also tested. If $\overline{\text{BRWCHAR}}$=0 more SAC contents have to be tested and control is returned to C3. Otherwise, testing of SAC has been completed and RF(3) will contain the recognized character code.

C5: LOAD CHAROUT

To load the character code into CHAROUT register the contents of RF(3) are passed to the F-BUS and CHAROUT is loaded at the end of the state.

C6: CLEAR IFP

The IFP counter is cleared preparing for the next cycle of the A-control.

C7: END-C

In this state the $\overline{\text{RTRN2}}$ signal goes low which makes the A-control proceed from A6 to all indicating a completion of a recognition cycle. The C-control will return to its WAIT state C_0 on the edge of the next Master Clock pulse.

3.17 The B-CONTROL

In the learning mode, the B-CONTROL is responsible for updating the CONFUMAT memory as well as the type of pointer (TP) and the pointer part (P-part) of the FEPOLIST memory. The details of the these updating operations will be considered in detail when describing the B-CONTROL flow chart. In the recognition mode, the CONFUMAT word, addressed by the P-part of FEPOLIST word, will be shifted bit by bit and added to the contents of the SCORING ACCUMULATOR registers (SAC). The control is then passed to the C-control to find the character code which obtains the maximum score in SAC. The following is the description of the B-CONTROL operations using the flow chart in Figure 3.21.

Note that the control signals of the B-control are active only when an $\overline{\text{ENB}}$=0 is received from the A-control in A6.

B0: READ1 CONFUMAT

When this state is active, the A-control is in A6. BCN is loaded with the addressed byte of the CONFUMAT in case of the recognition mode. In the learning mode BCN is cleared to prepare for inserting the appropriate character code in its corresponding bit position of the addressed byte. In the recognition mode, the control transfers to B1 while in the learning mode it transfers to either B9 or B12 according to the value of TP; either 1 or 0, respectively.

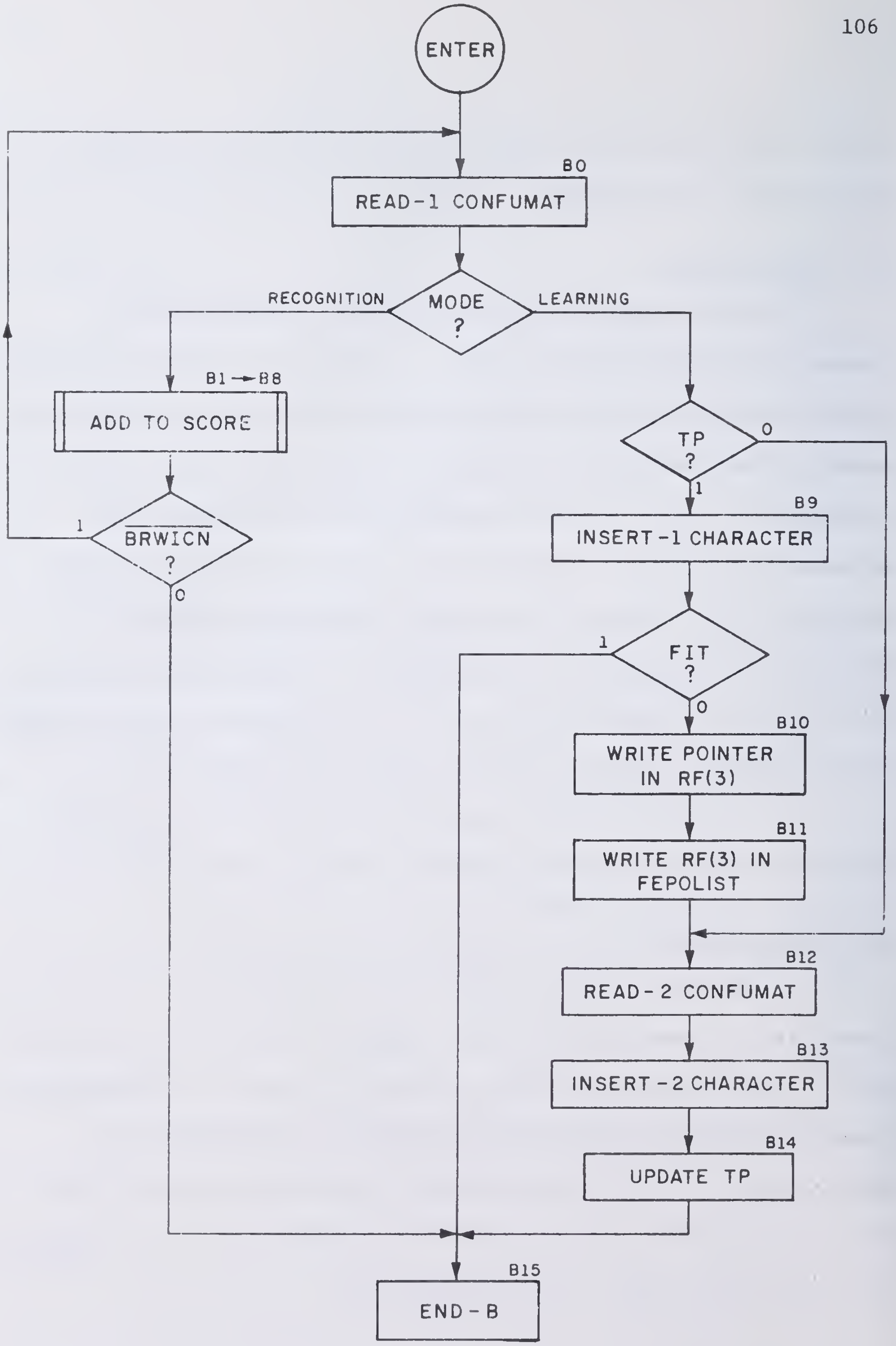


Figure 3.21 B-CONTROL Flow Chart

<u>B1→B8: ADD TO SCORE</u>

In this group of states BCN is shifted and its serial output is added to the previous contents of SAC. This process will be repeated by transferring back to B0 and entering this group until all the bits of the current CONFUMAT word have been added to the contents of SAC. Then, the control jumps to B15 (END-B).

<u>B9: INSERT1 CHARACTER</u>

The B-control branches to this state when the machine is in the learning mode and the information in the pointer byte of FEPOLIST is of the character code type (TP=1). FEPOLIST(3) and RF(3) are read and compared. If they are equal (FIT=1) this means that the input character code is the same as the stored character code and any other processing done in this state will be ignored. The control jumps to ENDB.

In case of FIT=0 more operations will be considered in the following control states beginning with B10. This state can be considered as a look-ahead state. The idea is to store the codes of the characters represented by FEPOLIST(3) and RF(3), in a new CONFUMAT word. In the current state the character code which resides in FEPOLIST(3) is put on the R-BUS and R0 and R2 are encoded into the buffer register BCN of CONFUMAT. This corresponds to the least significant 3 bits of the character code. The remaining most significant bits (R3-R4) are loaded into ICN to address the corresponding byte in the CONFUMAT word.

To summarize, the end result of B9 is a character encoded in ICN and BCN. As discussed before, the control will proceed to B10 if FIT=0.

B10: WRITE POINTER IN RF(3)

The address of the next available word of CONFUMAT is loaded from LC to MC. ICN and MC will address a single byte in CONFUMAT in which the contents of BCN will be written. The new pointer held in LC is also loaded into RF(3). On next clock the control proceeds to B11. TPF is also cleared in this state indicating that V-ELEMENT is not unique to a particular character anymore.

B11: WRITE RF(3) IN FEPOLIST

The new pointer is finally written into FEPOLIST(3). The update flip-flop (UPDF) is set to enable updating LCN later. The control proceeds to B12.

B12: READ2 CONFUMAT

The B-control enters this state either from B10 or from B0. The input character code, held in CHAR, is used to choose its corresponding byte in the addressed CONFUMAT word by loading ICN. This byte is read into BCN. The control proceeds to B13.

B13: INSERT2 CHARACTER

The new input character is encoded by setting its corresponding bit in BCN contents. IFP is also cleared preparing for updating the least significant bit of FEPOLIST(0).

B14: UPDATE TPF

To update FEPOLIST(0), RF(0) is read while its least significant bit is overridden by the contents of TPF. The F-BUS is then written in FEPOLIST(0). To prepare for updating LCN, MC is incremented at the end of this state.

B15: END-B

This is the end state of the B-control. The control may return to the A-control if the machine is in the learning mode. Otherwise, the C-control will be enabled. The LCN register is also updated by loading the contents of MC only when a new entry has been added in the CONFUMAT when UPDF contains 1 (see B11).

3.18 The A-CONTROL

Under the supervision of this control a request is issued to the V-control to construct a new V-ELEMENT and store it in the RF register file. When the control returns from V-CONTROL the FEPOLIST memory will be searched for a match between the contents of RF(0), RF(1) and RF(2) and the FEPOLIST F-part. At this point RF(3) will contain the input character code. If the machine is in the learning mode and no match was found, a new entry in FEPOLIST is generated (MARF is already pointing to the top of the memory stack). The type of pointer flip-flop (TPF) is set to contain '1' indicating a unique V-ELEMENT. This will be inserted in the first bit (bit number 0) of RF(0). In the new memory location in FEPOLIST the RF contents are written. If a match is found and the machine is in the learning mode the control is transferred to the B-CONTROL to complete the task of updating the DATA STRUCTURING memories.

In the recognition mode there are two possibilities. The first occurs when the newly constructed V-ELEMENT was not found in FEPOLIST. In this case the control goes back to V-CONTROL to construct another V-ELEMENT. This means that the current V-ELEMENT is ignored.

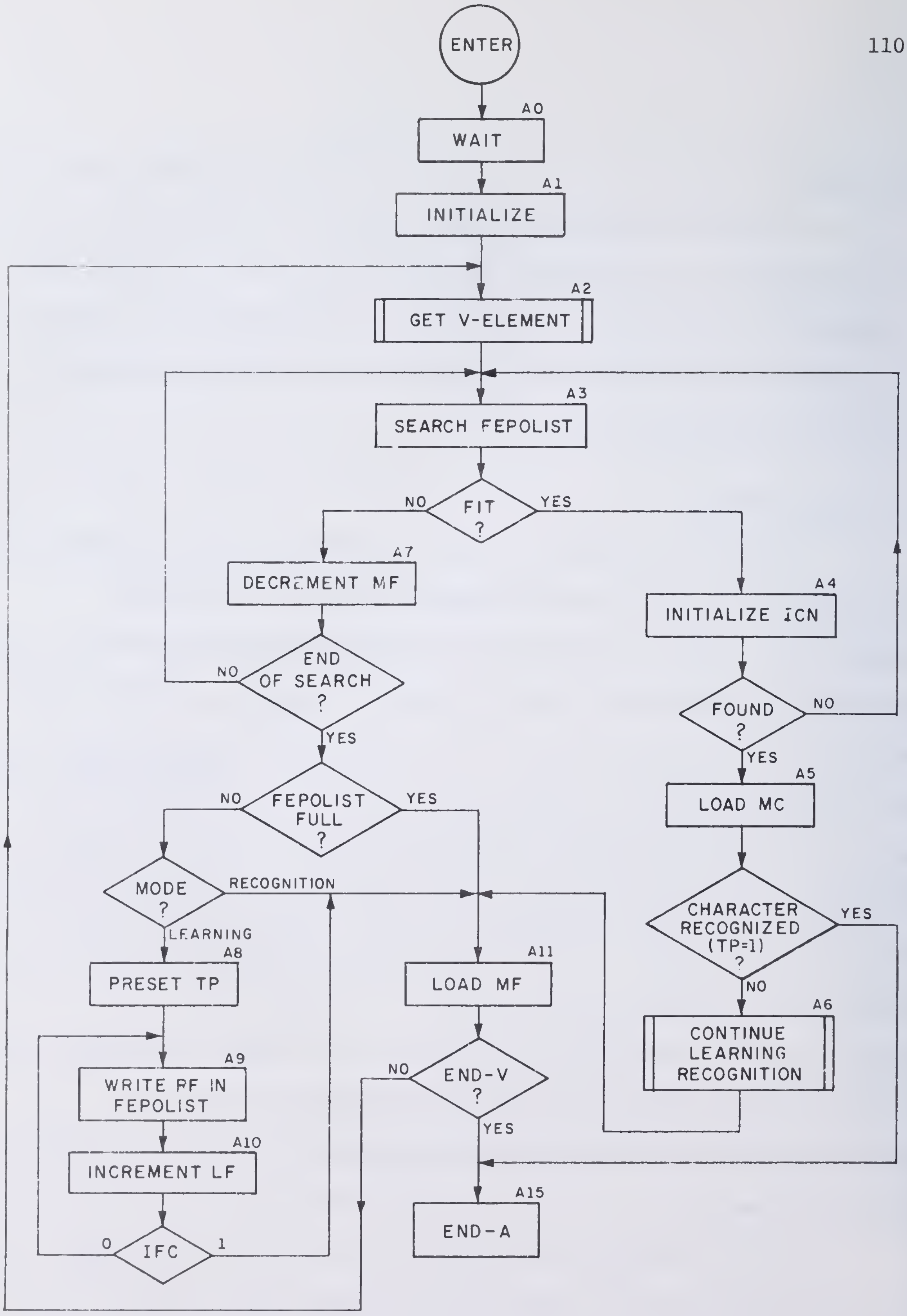


Figure 3.22 CLASSIFIER Operation Flow Chart (A-CONTROL)

The second possibility occurs when a match was found. If the corresponding TP in the matched FEPOLIST word is '1', this means the current V-ELEMENT corresponds to a unique character code which will be considered as the recognized character. Otherwise, when TP=0, the control transfers to the B-CONTROL for further processing.

The above is a summary for the operation of the A-CONTROL the details are discussed below with the aid of the flow chart in Figure 3.22.

A0: WAIT

This is the reset state of the A-control. The A-control will stay in this state until an enable signal ($\overline{\text{ENA}}$=0) has been received from the M-CONTROL ($\overline{\text{ENCLASFY}}$=0). Upon receiving this signal the A-control goes to the INITIALIZE state A1.

A1: INITIALIZE

Three operations are performed in this state: loading RF by the input character code, initializing the memory address register MF of FEPOLIST and clearing the scheme address register/counter AV of the V-ELEMENT CONSTRUCTION module. The first operation is accomplished by loading IFP with 1's, enabling the output of CHAR counter and writing the CHAR contents into the addressed location in RF. This will place the input character code in RF(3) in the case of learning mode. In the recognition mode the initial contents of RF(3) will be ignored.

The MF register/counter is loaded with the contents of LF which holds the address of the next available location top of the stack of the FEPOLIST memory. AV is cleared to begin processing of the FEATURE VECTOR with the first scheme of VMEM.

A2: GET a V-ELEMENT

A request is issued to the V-control to get a new V-ELEMENT by holding down the $\overline{\text{ENV}}$ line. This signal will make the V-control transfer from the WAIT sate (V0) once it gets there. When returning from the V-control the RF registers RF(0), RF(1) and RF(2) will hold a V-ELEMENT. The A-control will then proceed to state A3.

A3: SEARCH FEPOLIST

This state and the states A7 and A4 can be viewed as the SEARCHING states. In state A3 a match is tried between the currently addressed register in RF and the elements of the corresponding bytes of FEPOLIST. If a match is found the transfer of control goes to A4. Otherwise, the transfer goes to A7. The TPF flip-flop is always loaded with the least significant bit of the R-BUS when IFP=0. The IFP counter is incremented at the end of this state so that the next byte of the V-ELEMENT will be tried if the control comes back to this state.

A4: INITIALIZE ICN

ICN is loaded with all 1's to prepare for addressing the most significant byte of CONFUMAT in case the control transfers to the B-CONTROL. At the end of this state IFP is tested for completion of matching V-ELEMENT in RF against the contents of FEPOLIST. If the matching has been completed successfully (IFP=3) the control proceeds to A5. Otherwise, the control is transferred back to A3 with a new IFP in order to find a match to the remaining bytes of V-ELEMENT.

A5: LOAD MC

The pointer byte, is read from FEPOLIST (3) and loaded into MC. If the device is in the recognition mode the pointer byte is also loaded into RF(3) and passed to the F-BUS ($\overline{\text{RRF}}$=0) to be loaded into CHAROUT ($\overline{\text{LDCHAROUT}}$=0) at the end of A5. In case of TP=1, this will be the recognized character code, otherwise it will be rewritten in subsequent cycles with the recognized character code. If TP=1 and we are in the recognition mode the current classification ends by transferring the control to END-A state. If TP=0 further processing is needed for both learning and recognition modes. The control proceeds to A6 enabling the B-CONTROL.

A6: CONTINUE LEARN/RECOGNITION

The A-CONTROL remains in this state until a completion code has been received from the B-control or the C-control. When this happens the control jumps to state All.

A7: DECREMENT MF

If the matching has failed at the end of state A3 (FIT=0) the control trnsfers to this state. Here the memory address register MF is decremented and the IFP counter is cleared. This will insure that another FEPOLIST word will be tested and the testing will begin by the first byte of the FEPOLIST new word and RF(0). If the last word of FEPOLIST has been tested (MF=0) the $\overline{\text{BRW}}$ line of MF will go to 0 and branching occurs on the next clock pulse to either A8 or All. If the machine is in RECOGNITION mode (LRN=0) or the FEPOLIST memory is full (FPFULL=1) the next state will be All. Otherwise, the next state will be A8.

A8: PRESET TPF

This state can be viewed at as an initialization state for the combination of A9 and A10, which are responsible for storing a new entry in FEPOLIST. To do that, MF is loaded with the address of the first available memory location (the top of stack) from the LF register. To indicate that this entry is unique the TP flip-flop preset to 1.

A9: WRITE RF IN FEPOLIST

The content of TPF is passed to F0 of the F-BUS when writing RF(0) into FEPOLIST. In the subsequent passes of A9 the contents of RF(1), RF(2), RF(3) will be written in FEPOLIST (1), (2), (3), respectively. At the end of the state IFP is incremented to address the next byte of FEPOLIST and the corresponding register of RF.

A10: INCREMENT IF (STACK POINTER)

LF is updated by incrementing MF, which contains the current value of LF, and loading MF back into LF. This is done provided that the FEPOLIST is not full. At the end of A10 the contents of IFP is tested for completion of storage operation. When IFP is 4 the operation has been completed and control proceeds to A11. Else, the control transfers back to A9.

A11: LOAD MF

In this state MF is loaded with the address of the next available memory location (LF) in FEPOLIST. If there are more V-ELEMENTs to be extracted and processed ($\overline{\text{ENDV}}$=1) the next state will be A2. In case of $\overline{\text{ENDV}}$=0 the classification cycle (LEARNING or RECOGNITION) is completed and the next state will be ENDA ($\overline{\text{ENDLASFY}}$=0).

A15: END-A ($\overline{\text{ENDCLASFY}}$=0)

The classification cycle ends here and the control transfers back to the M-CONTROL.

4. SUMMARY AND CONCLUSIONS

A trainable, real-time character recognition device has been designed and built. The visual feature extraction method was chosen as the most favorable computational approach around which the feature extractor was designed. Human factors were considered in choosing the features and the two-dimensional, binary input form was exploited in devising methods for preprocessing the character image and extracting the local features.

A method of segmentation was introduced, in which the character is described as a collection of either horizontal or verticle segments connected by links. Segments may consist of several disconnected subsegments, each of which may contain one or more local features (nodes). Links connect nodes of a segment to neighboring segments. For the current implementation, minimum descriptions of the constituent parts of the character--segments, links and nodes--were considered. The existence of segments and subsegments, the number of links and their distribution within subsegments and the number of nodes in each subsegment constitute the feature vector. This multiple description technique made it possible to reduce the number of training samples and to use a simple data structure for the classification dictionary as well as a simple maximum fit decision criterion.

The introduction of compound features (V-ELEMENTS) allowed the number of the training samples to be reduced still further.

Moreover, the programmability of the V-ELEMENT construction schemes gives the user more flexibility in enhancing the recognition rate by choosing the feature combinations and writing their schemes in the writable memory VMEM.

To reduce the storage required for the classification dictionary, the DATA STRUCTURING part of the CLASSIFIER was designed to include some general properties that deal with the relationships between features (V-ELEMENTS) and character classes. A V-ELEMENT may appear in several character classes which can be grouped and encoded as a CONFUMAT word addressed by a pointer associated with the V-ELEMENT in FEPOLIST. In this way, the storage of the true character codes were avoided which resulted in reduction of storage. On the other hand, if a V-ELEMENT is unique to a certain character a flag bit will indicate that the pointer part is the character code. In this case a CONFUMAT word is saved and the recognition is speeded up. The CLASSIFIER memories, CONFUMAT and FEPOLIST, were also designed to make it easy to add new character classes and even to introduce new features not detected in previous character samples. CONFUMAT can be expanded horizontally, bit-wise, to accommodate more character classes and FEPOLIST can be expanded vertically, word-wise, to store more features. In addition, the encoding of character classes as CONFUAMT words made it possible to implement simple maximum fit decision criterion.

The input image as well as the extracted features are stored in two-dimensional registers (WORKING STORAGE) organized in such a way as to be easily accessed in both the x and y directions. This organization, together with the facility of scanning made the preprocessing of the image plane and the extraction of different features extremely straightforward, easy to implement and reasonably fast.

Minimal size, 3x3 patterns were used not only in preprocessing the input image but also in extracting the nodes. Table look-up techniques were used to perform these operations by using ROMs for more efficient storage of the necessary patterns and for processing speed up.

Because of the way people draw line-like patterns, and because of quantization noise and the small size of the windows used in node extraction, not all the extracted nodes are necessary for representing the drawn character. MERGING RULES were therefore devised for operating on node memory contents to eliminate superfluous nodes and modifying others. The careful selection of these rules enabled them to be applied to only two neighboring cells at a time using a special scanning facility and ROM addressing while scanning the WORKING STOAGE.

Nodal extraction using 3x3 windows combined with the merging operations on two neighboring nodal cells made an elegant, but nevertheless powerful method for detecting local features in line-like patterns. It should also be noted that all the operations involved in preprocessing and feature extraction as well as constructing the feature subvectors are done by simple raster scanning of the WORKING STORAGE memories without resorting to contour following or backtracking. In addition, no mathematical computations were involved in any of these operations other than simple ROM addressing using small size memories. These techniques of simple raster scanning and

ROM addressing simplified the control and rendered the implementation more elegant than in the other approaches described earlier.

Because of the unavailability of a writing tablet and hardware reliability problems, statistical studies could not be made. Otherwise, detailed data could have been obtained on the effect on the recognition rate of various parameters, e.g. character classes and their numbers, number of training samples per class, V-ELEMENT construction schemes, etc. However, the device as it stands now shows the validity of the general approach and the power of new methods of processing and hardware organization.

Immediate enhancements could be made to the device to achieve a higher recognition rate. One is adding a module which forms the local feature subvector NODEV and changing the V-ELEMENT construction schemes to reflect that. Expanding the FEPOLIST memory might also be desirable to accommodate more feature compounds and hence more character classes.

Currently the construction schemes are tried separately for horizontal or vertical segmentation. Possible improvement would be to have new construction schemes which construct compounds of features from different horizontal and vertical segments. This would require both wider VMEM and FEPOLIST words. The decision criterion was based on simple matching and counting the number of successful matchings.

Another improvement might be the introduction of higher level feature compounds consisting of combinations of the previously addressed pointers. These new compounds would be stored in turn with a new set of pointers to CONFUMAT. This process can be iterated to any higher level wanted. The technique could enhance the recognition rate tremendously and reduce the training samples still further. Another improvement could be the addition of an EXCEPTION MATRIX, organized similarly to CONFUMAT, to be used in erasing the mistakes otherwise made in recognition.

At the time this project was envisioned (1973), large scale integrated (LSI) chips, such as microprocessors and bit-sliced microprocessors, were not readily available and, even two years later, were rather expensive. But now, in 1978, they are available at affordable prices. If one were to redesign the whole system, use would be made of these LSI chips and others, e.g. programmed logic arrays, in implementing the device. For example, microsequencers and PLAs could be used instead of the random logic circuits which were used in implementing the controls of the preprocessor and the feature extractor. The MASTER control could be implemented as a microcomputer and a slave microcomputer could take over the functions of the CLASSIFIER. However, because of the current state of the art of technology, speed and cost, this system would be slower and more expensive than the current system. On the other hand, reliability would be higher and the system would gain in terms of versatility and flexibility. But, the shift in implementation from hardware to software might restrict the ability to investigate architecture more suitable to the character recognition problem.

Still further in the future is the prospect of VLSI content addressable memories. This type of architecture would be ideal for the pattern recognition problem in general including character recognition. Notice, for example, the CLASSIFIER and how its operations, search and insert by content, are exactly the operations of content addressable memories. If only a part could be implemented directly in LSI, it would be the WORKING STORAGE. This is because of its simple regular cell structure. One might also envision a WORKING STORAGE implemented with CCD technology as an integral part of the retina of a CCD camera.

Besides meeting the challenge of implementing a trainable, real-time character recognition machine, this project has introduced new powerful techniques which have opened the way for future improvements and for new possibilities in designing a character recognition machine.

REFERENCES

[1] Harmon, L. D., "Automatic Recognition of Print and Script," Proc. of IEEE, Vol. 60, No. 10, pp. 1165-1176, Oct. 1972.

[2] Rosenfeld, A., "Picture Processing: 1974," Computer Graphics and Image Processing, Vol. 4, No. 2, pp. 133-155, June 1975.

[3] ________, "Picture Processing: 1975," Computer Graphics and Image Processing, Vol. 5, No. 2, pp. 215-237, June 1976.

[4] ________, "Picture Processing: 1976," Computer Graphics and Image Processing, Vol. 6, No. 2, pp. 157-183, April 1977.

[5] Miller, G. M., "On-line Recognition of Hand-generated Symbols," FJCC, 1969, pp. 399-412.

[6] Kwon, S. K. and Lai, D. C., "Recognition Experiments with Hand-printed Numerals," 1976 Joint Workshop on Pattern Recognition and Artificial Intelligence, pp. 74-83.

[7] Bledsoe, W. W. and Browning, I., "Pattern Recognition and Reading by Machine," 1959 Easter Joint Computer Conference, pp. 225-232.

[8] Fairhurst, M. C. and Stonham, T. J., "A Classification System for Alphanumeric Characters Based on Learning Network Techniques," Digital Process 2 (1976) 3321.

[9] Young, T. Y. and Calvert, T. W., Classification, Estimation and Pattern Recognition, American Elsevier, New York (1974).

[10] Wendling, S. and Stamon, G., "Hadamard and Haar Transforms and Their Power Spectra in Character Recognition," 1976 J. Workshop on P.R. and A.I., pp. 103-112.

[11] Tucker, N. D. and Evans, F. C., "A Two-Step Strategy for Character Recognition Using Geometrical Moments," Second International Joint Conference on Pattern Recognition, August 1974, pp. 223-225.

[12] Berthod, M. and Maroy, J. P., "Morphological Features and Sequential Information in Real-time Handwriting Recognition," Second International Joint Conference on Pattern Recognition, August 1974, pp. 358-363.

[13] Watt, A. H. and Beurle, R. L., "Recognition of Handprinted Numerals Reduced to Graph-representable Form," Second Intl. Joint Conf. on A.I., September 1971.

[14] Attneave, A., "Some Informational Aspects of Visual Perception," Psychological Review, No. 61, pp. 183-193 (1954).

[15] Genchi, H., Mori, K., Watanable, S. and Katasuragi, S., "Recognition of Handwritten Numerical Characters for Automatic Letter Sorting," Proc. IEEE, Vol. 56, No. 8, pp. 12992-1301, August, 1968.

[16] Karpinski, J. and Michalski, R., "A Recognition System for Alphanumeric Characters," Proceedings of the Institute for Automatic Control, Polish Academy of Sciences, No. 35, 1966, Warsaw (in Polish).

[17] Freeman, J., "Survey - The Modelling of Spatial Relations," Computer Graphics and Image Processing, Vol. 4, No. 2, pp. 156-171, June, 1975.

[18] Arps, R. B., "Entropy of Printed Matter at the Threshold of Legibility for Efficient Encoding in Digital Image Processing," Technical Report No. 69-36251, Stanford University, 1969.

APPENDIX A

OPERATION AND EXPERIMENTS

A.1 Initialization

This is the process of initializing the writable memories (FEPOLIST and CONFUMAT) of the DATA STRUCTURING part of INCOM CLASSIFIER. This process is done after turning on the machine or reprogramming the V-ELEMENT CONSTRUCTION memory (VMEM), or when changing the character set to be recognized by the machine. CONFUMAT is initialized by clearing its contents, by setting the INPUT switches to 0's and the INIT-C switch on the B-CONTROL card to CLR-C position. Similarly, to initialize FEPOLIST the INPUT switches are set to 0's and INIT-FP switch on the A-CONTROL card is set to CLR-F position. In either case, setting the INIT switch to the CLR position will enable corresponding circuits on the A-CONTROL or the B-CONTROL cards to produce the necessary signals to increment memory address counters and the index registers as well as writing pulses to the corresponding memories. This will result in addressing the memory bytes sequentially and writing the INPUT switches data in them.

A.2 Programming VMEM

This is the process of writing new V-ELEMENT CONSTRUCTION SCHEMES in VMEM or modifying old ones. The contents of VMEM (16 bytes) can be examined or rewritten one byte at a time. As in the INITIALIZATION process, the PROGRAMMING is done when the machine is in the INITIALIZE state (M1). To program VMEM the OPERATION SELECTION switch is set in the PROGRAM position. To examine a VMEM byte, the INPUT switch must be in the ADDRESSES position and the right most four switches of the INPUT group is set to indicate the byte address. Upon depressing the DEPOSIT switch the contents of the examined byte will be displayed in the DATA OUT group. To rewrite the contents of a byte, the examine step is performed first followed by setting the INPUT switch to DATA and positioning the DATA INPUT switches to reflect the new data. To complete the writing step, the DEPOSIT switch is depressed.

The signals necessary for the above operations are generated on the CLOCK and PROGRAMMING module (see Appendix B).

A.3 Drawing Characters

Under the STORE CONTROL LOGIC (S-CONTROL) (see Appendix B) INCOM can accept a series of points either automatically by communicating with a tablet-digitizer or manually by the user in form of ENTER! commands. In either case the machine must be in DRAW state (M2) and the CONTACT/END switch in the CONTACT position. To terminate the drawing operation the switch is set in END position. Each accepted point is stored in its corresponding cell in IMAGEM, which contents appear on the CRT display during the DRAW state. Because of the unavailability of a graphics tablet, only the manual form of input is used using the SIMULATED TABLET SIGNALS switch group.

When X-Y COORDS LED is on, a curser point appears on the CRT screen. The curser scans contents of IMAGEM horizontally. It can also be positioned on any horizontal line by pressing the DEPOSIT button. To input a point the user presses the ENTER! switch when the curser coincides with the required point position on the screen.

To input a character the user can follow the following steps:

i. Draw the character on a transparent sheet.

ii. Overlay the sheet on the CRT screen and trace the drawing point by point by pressing ENTER! switch whever the curser coincides with the drawing.

iii. When the drawing process is finished position the CONTACT/END switch in the END position. This will let the machine begin processing the input image.

A.4 Example of V-ELEMENTS Construction Schemes

As discussed before V-ELEMENTS are constructed using construction schemes which have been written in VMEM while initializing the machine. Because of the experimental nature of INCOM, we have chosen to let VMEM programming accessible to the user. We may recall that the construction schemes are used in both scanning modes. We may also recall that a construction scheme is stored as two bytes in VMEM; each byte will be used in constructing a byte of the feature part of the V-ELEMENT. In other words each feature byte can be constructed independently of the other one. But we have to remember that TC or BC will remain the same for either byte. Furthermore, the V-CONTROL was designed in such a way that TC will scan half of the character representation from one end while BC scans the other half from the opposite end.

The construction schemes can vary from the simple to the complex and from the general to the more specific. A simple scheme may result in constructing a V-ELEMENT consisting of elements of different feature subvectors which describe a particular segment. A more complex scheme results in combining several elements from one subvector but belonging to two segments addressed by the top (left) segment counter (TC) and the bottom (right) segment counter (BC).

Since no more than eight different schemes can be stored in VMEM, only schemes which combine two or more subvectors are considered. A possible set of construction schemes are shown in Table A.1. They are ordered hierarchically from the general to the specific and from the simple to the complex. The first four construction pairs construct V-ELEMENTS using elements from different subvectors of the same segment addressed by either TC or BC counters (used as subscripts in the table). The first two pairs use the LINKV subvectors, ILINKV and OLINKV. The first pair scans the character from the top (left) using TC while the second scans it from the bottom (right) using BC. The next two pairs use the three subvectors, ILINKV, OLINKV and TABV. Hence, the resulting V-ELEMENTS are more complex and more specific than the previous. The remaining four pairs will construct more complex and more specific V-ELEMENTS than the first four. They construct V-ELEMENTS using two or more subvectors of two different segments, one addressed by TC and the other by BC. The first pair of the last four uses the LINKV subvectors, the second pair uses ILINKV and TABV and the third uses OLINKV and TABV. The most complex of them all is the last pair which constructs V-ELEMENTS using OLINKV and TABV of a top (left) segment and ILINKV and TABV of a bottom (right) segment.

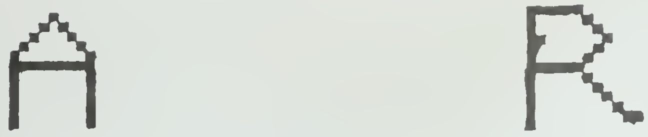
Table A.1

V-ELEMENTS CONSTRUCTION SCHEMES

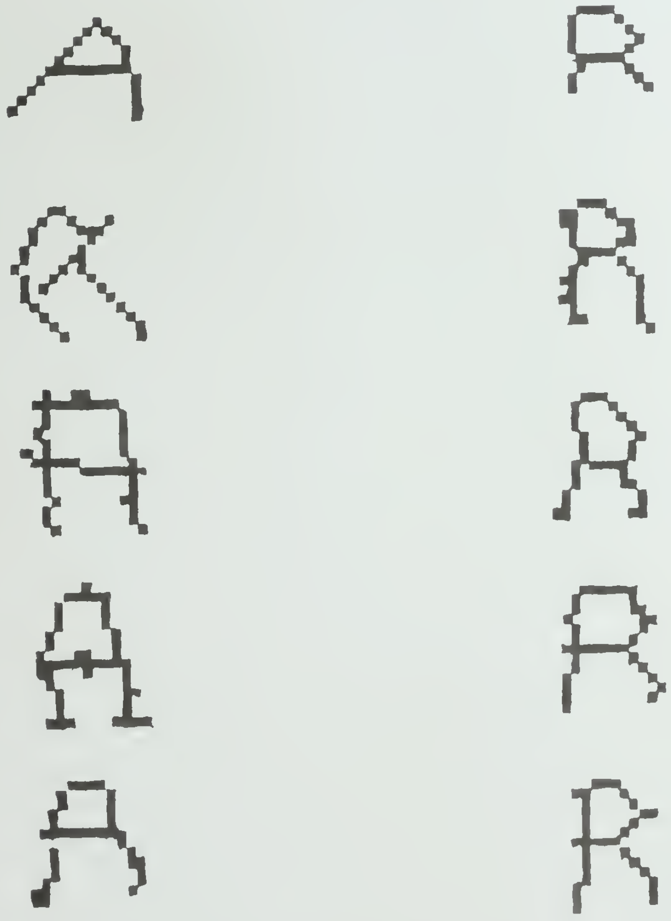
ADDRESSES	DATA	SCHEMES
0	0100 0100	OLINKV(TC), ILINKV(TC)
1	0100 0100	~ ~
2	0100 1100	OLINKV(BC), ILINKV(BC)
3	0100 1100	~ ~
4	0100 0100	OLINKV(TC), ILINKV(TC)
5	0010 0010	TABV(TC), TABV(TC)
6	0100 1100	OLINKV(BC), ILINKV(BC)
7	0010 1010	TABV(BC), TABV(BC)
8	0100 0100	OLINKV(TC), ILINKV(TC)
9	0100 1100	OLINKV(BC), ILINKV(BC)
10	0010 0100	TABV(TC), ILINKV(TC)
11	0010 1100	TABV(BC), ILINKV(BC)
12	0100 0010	OLINKV(TC), TABV(TC)
13	0100 1010	OLINKV(BC), TABV(BC)
14	0100 0010	OLINKV(TC), TABV(TC)
15	0010 1100	TABV(BC), ILINKV(BC)

A.5 Experiments in Recognition

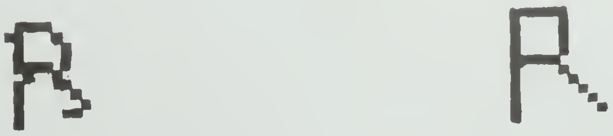
After programming VMEM with the construction schemes of Table A.1 the machine was trained to recognize two different classes 'A' and 'R' using only one training character for each class (Figure A.1.a). Several testing characters were presented to INCOM for recognition. Examples of the correctly recognized characters and misrecognized are shown in Figure A.1. The testing characters were intentionally drawn distorted and noisy to show the recognition power of INCOM. In real-life situation characters the user draws should not be of such variety once he trained the machine to recognize his characters. The two misrecognized characters were intended to be members of class 'R'. However, looking closely these characters might also be misrecognized by humans as 'A's. Although one training character was used for each class, INCOM recognized quite a variety of character shapes using only the three subvectors ILINKV, OLINKV and TABV.



a) Training Characters



b) Recognized Characters



c) Misrecognized Characters ('A' instead of 'R')

Figure A.1 Example of Recognition

APPENDIX B

INCOM CIRCUIT DIAGRAMS

LIST OF FIGURES

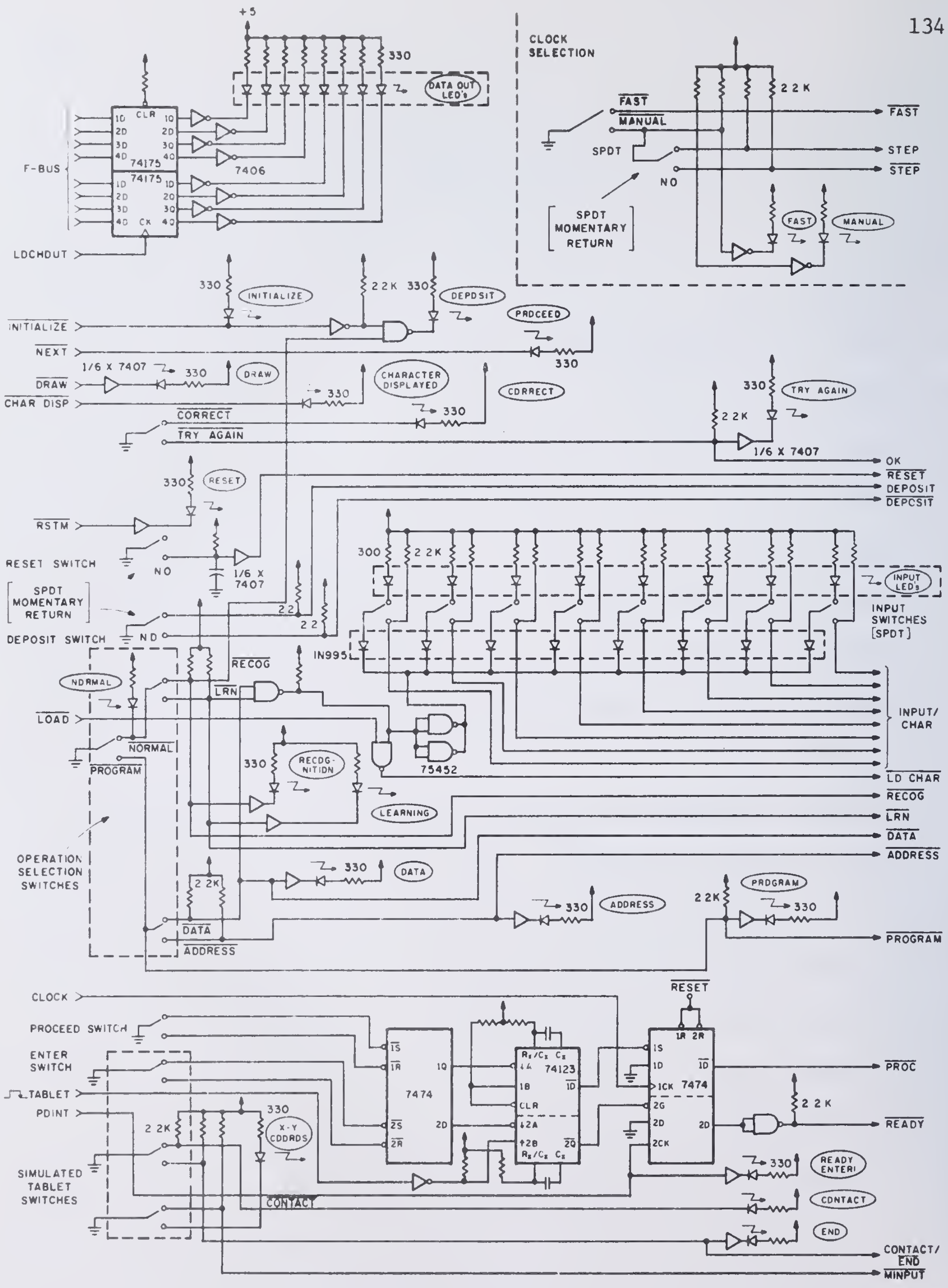


Figure B.1 The PANEL Cards

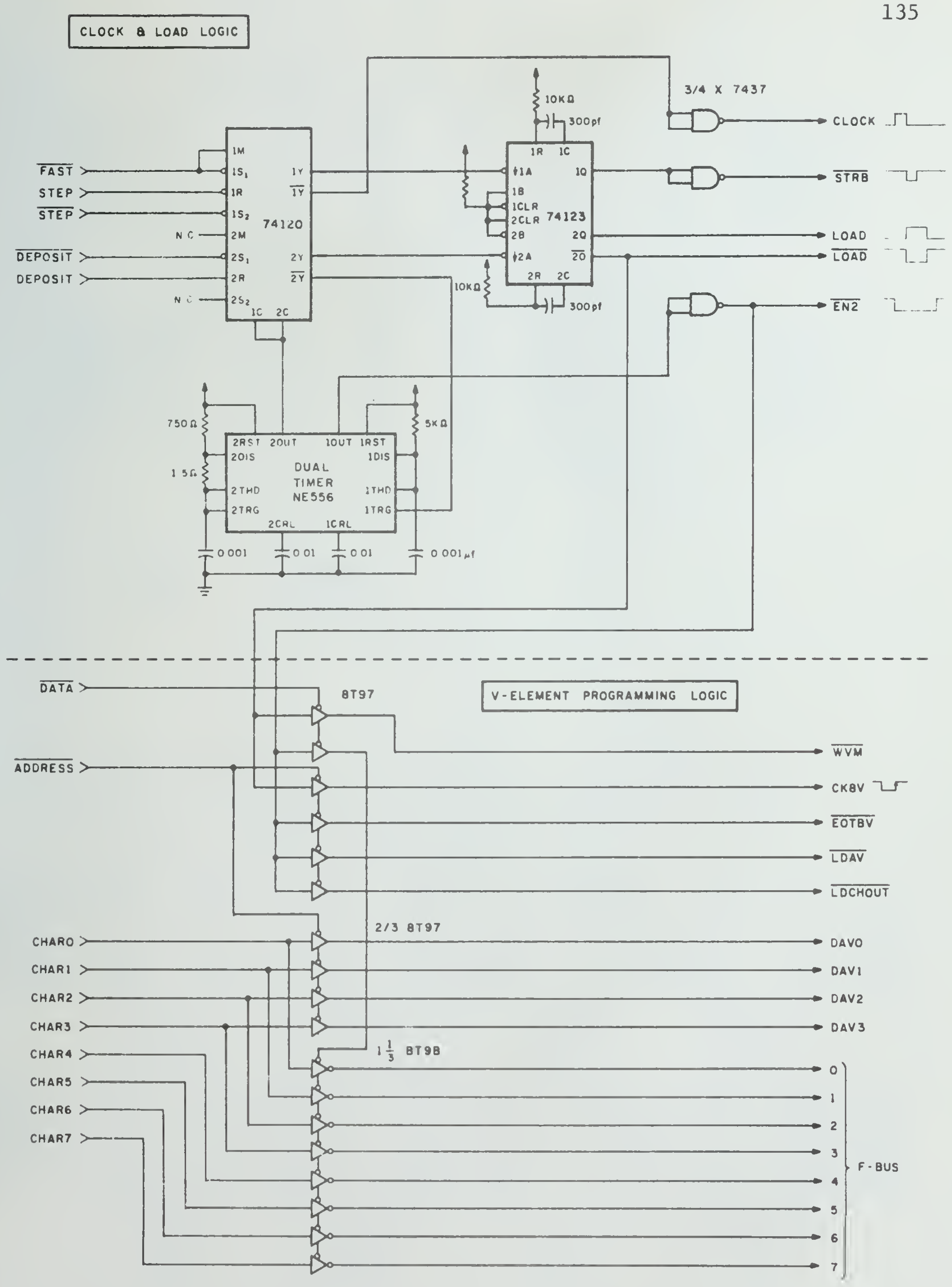
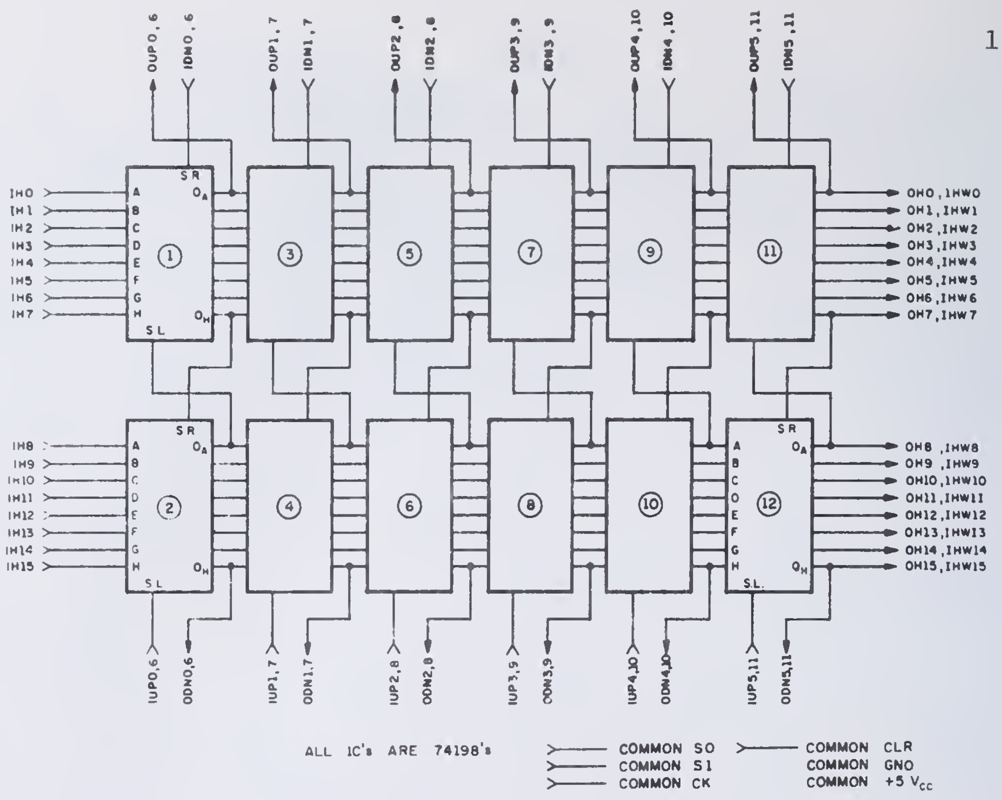
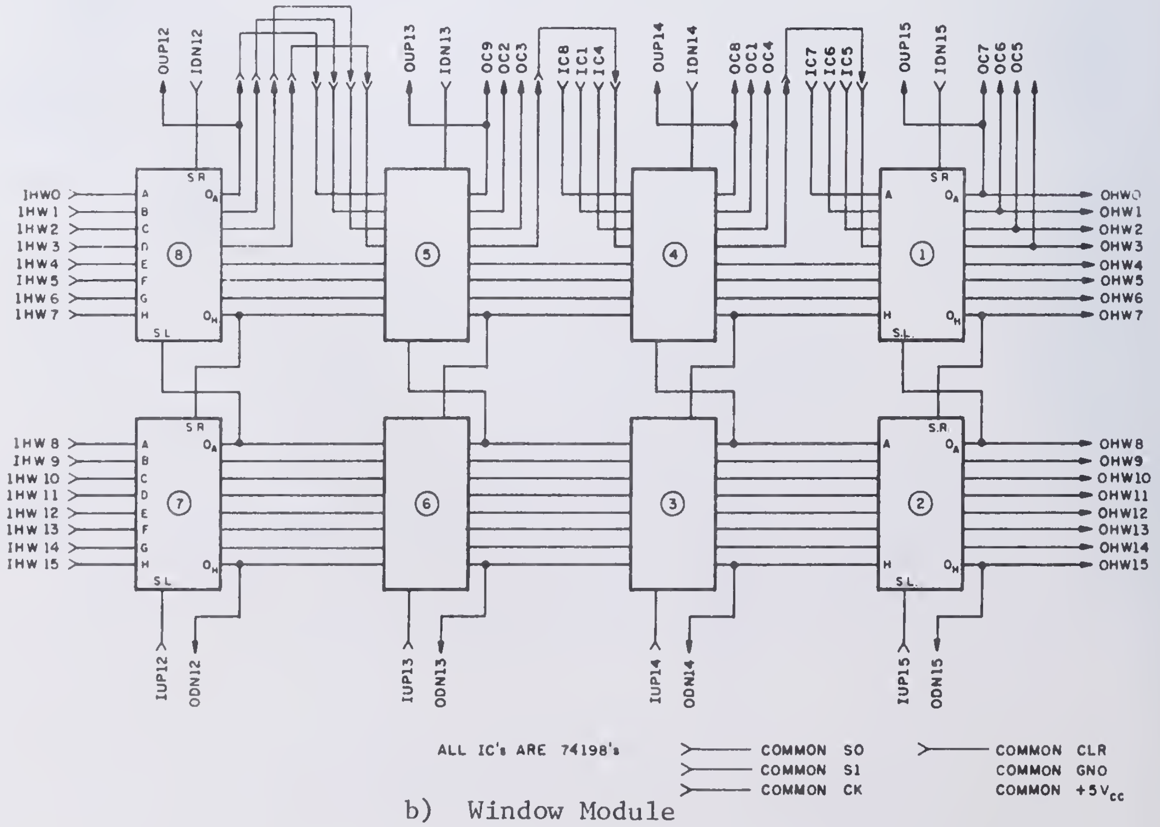


Figure B.2 CLOCK and PROGRAMMING Card



a) First (Second) Module



b) Window Module

Figure B.3 WORKING STORAGE Plane

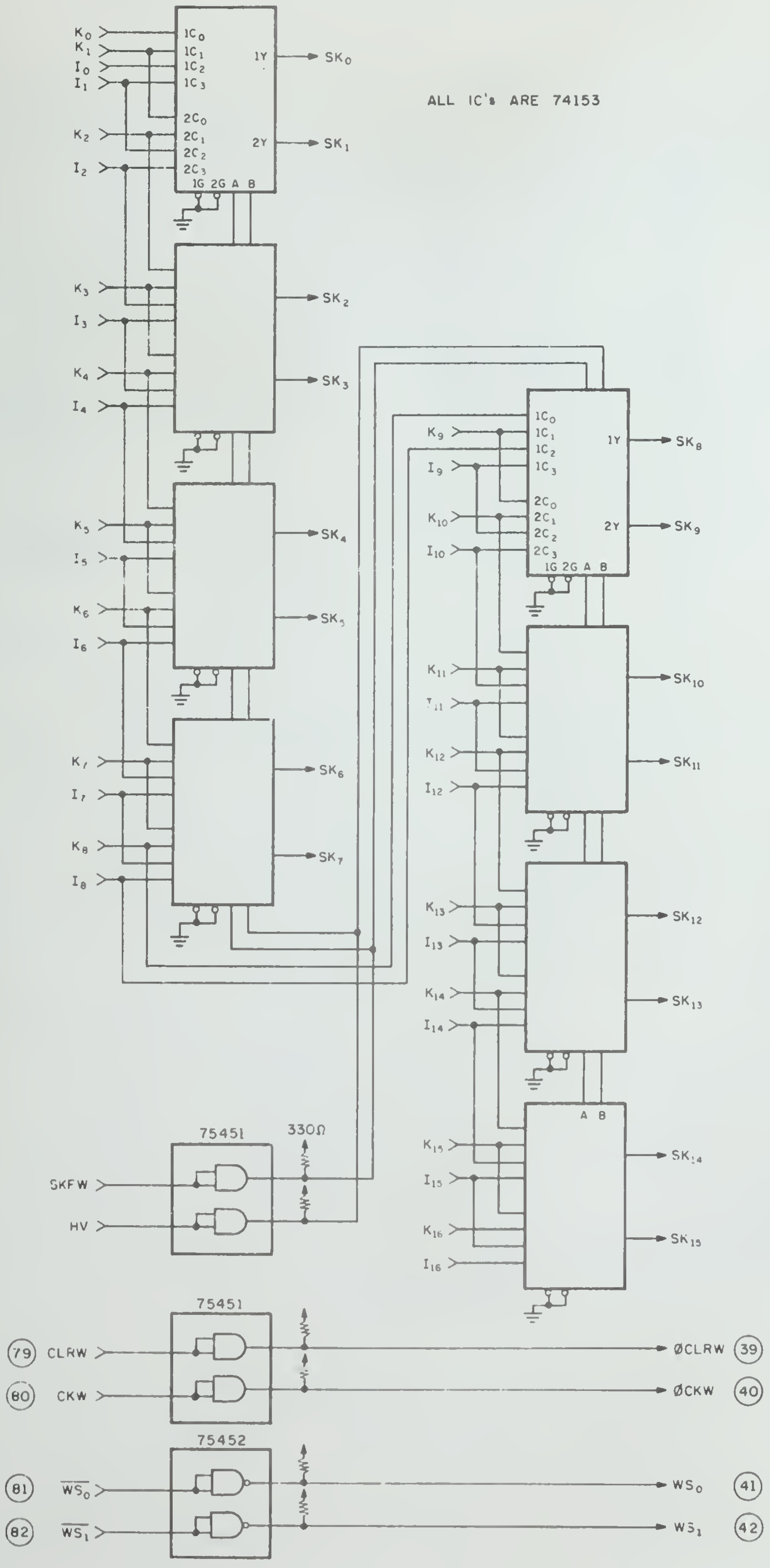


Figure B.4 SKEW/TRANSFER Module

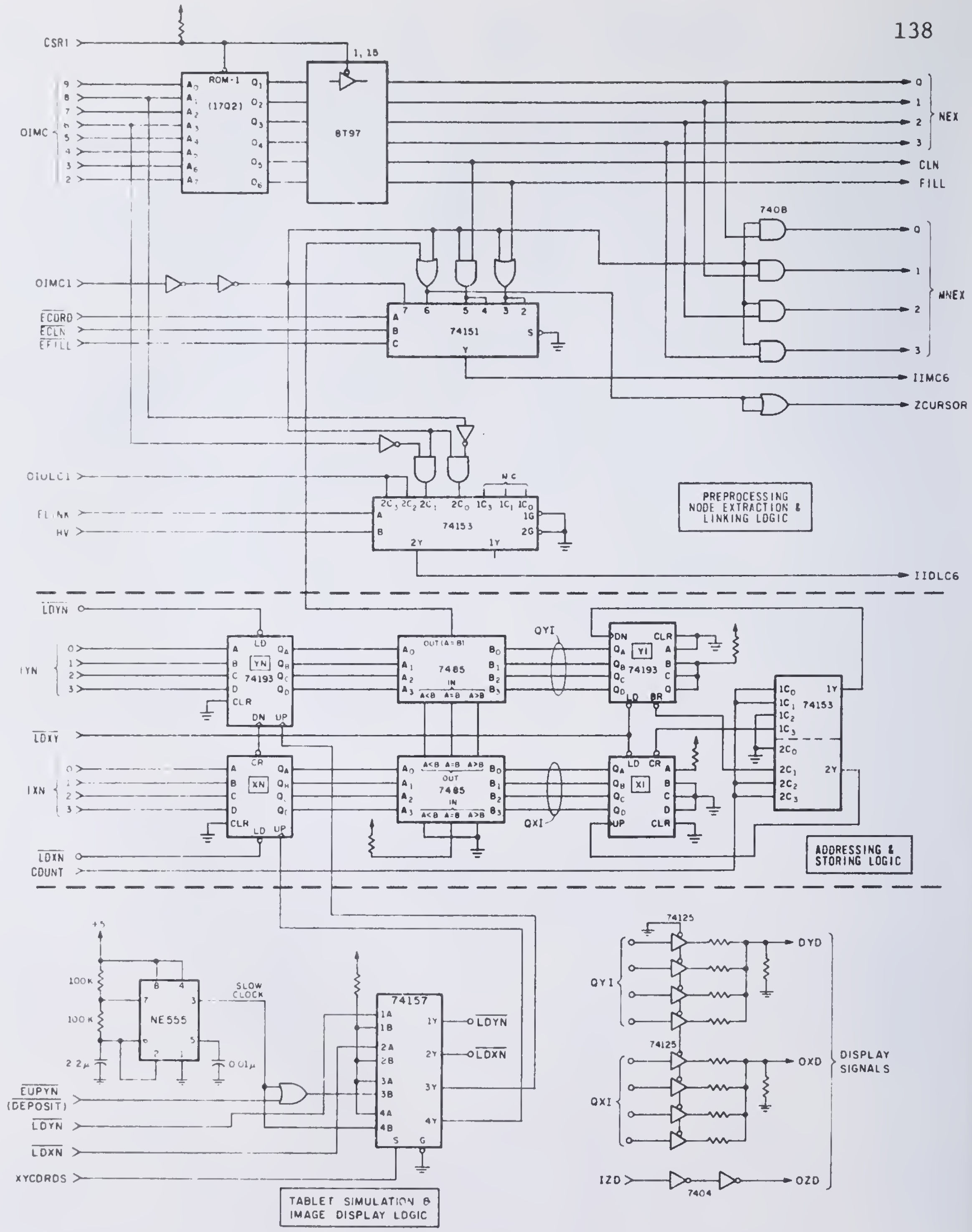


Figure B.5 PRELOG Module

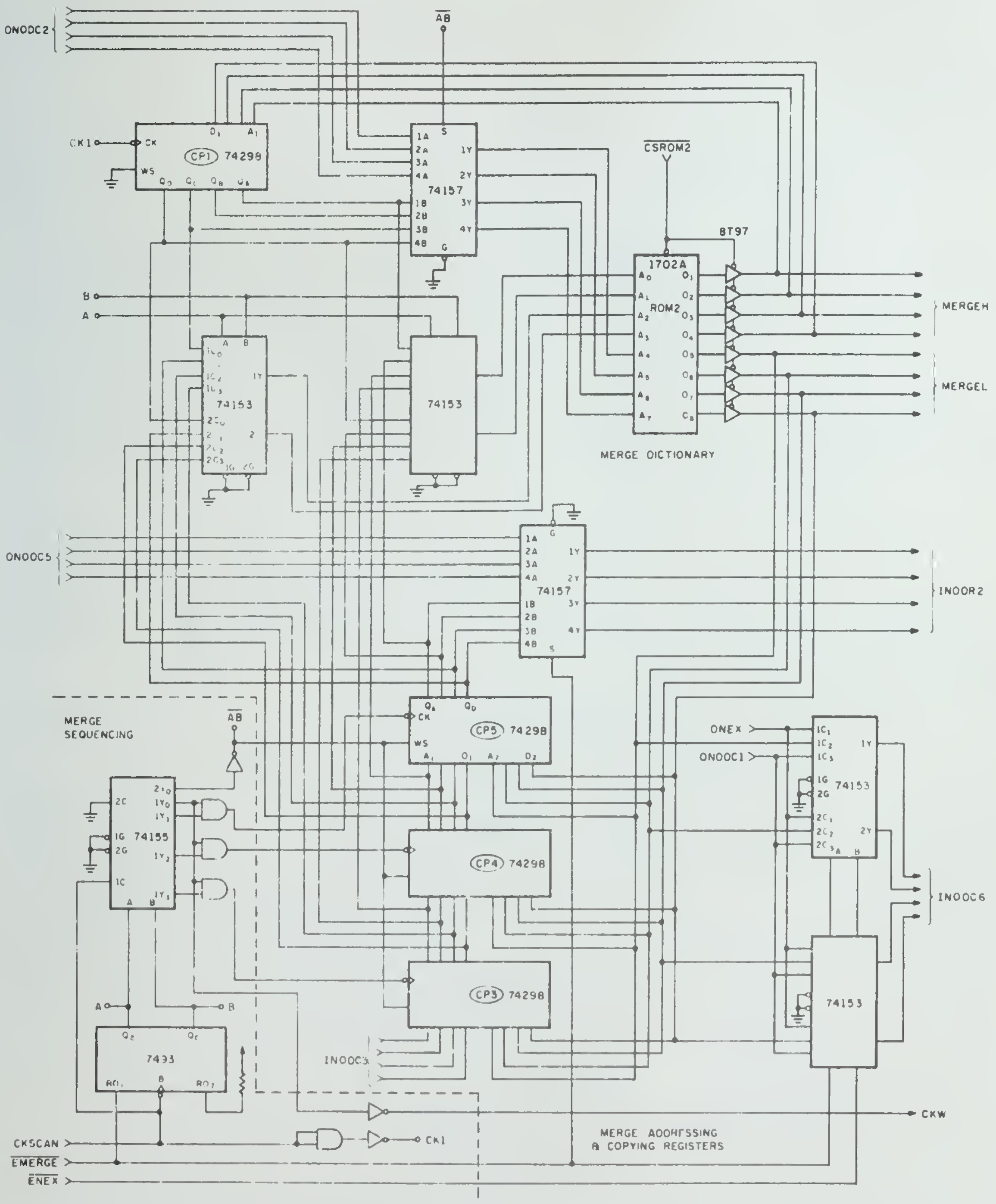


Figure B.6 MERGER Module

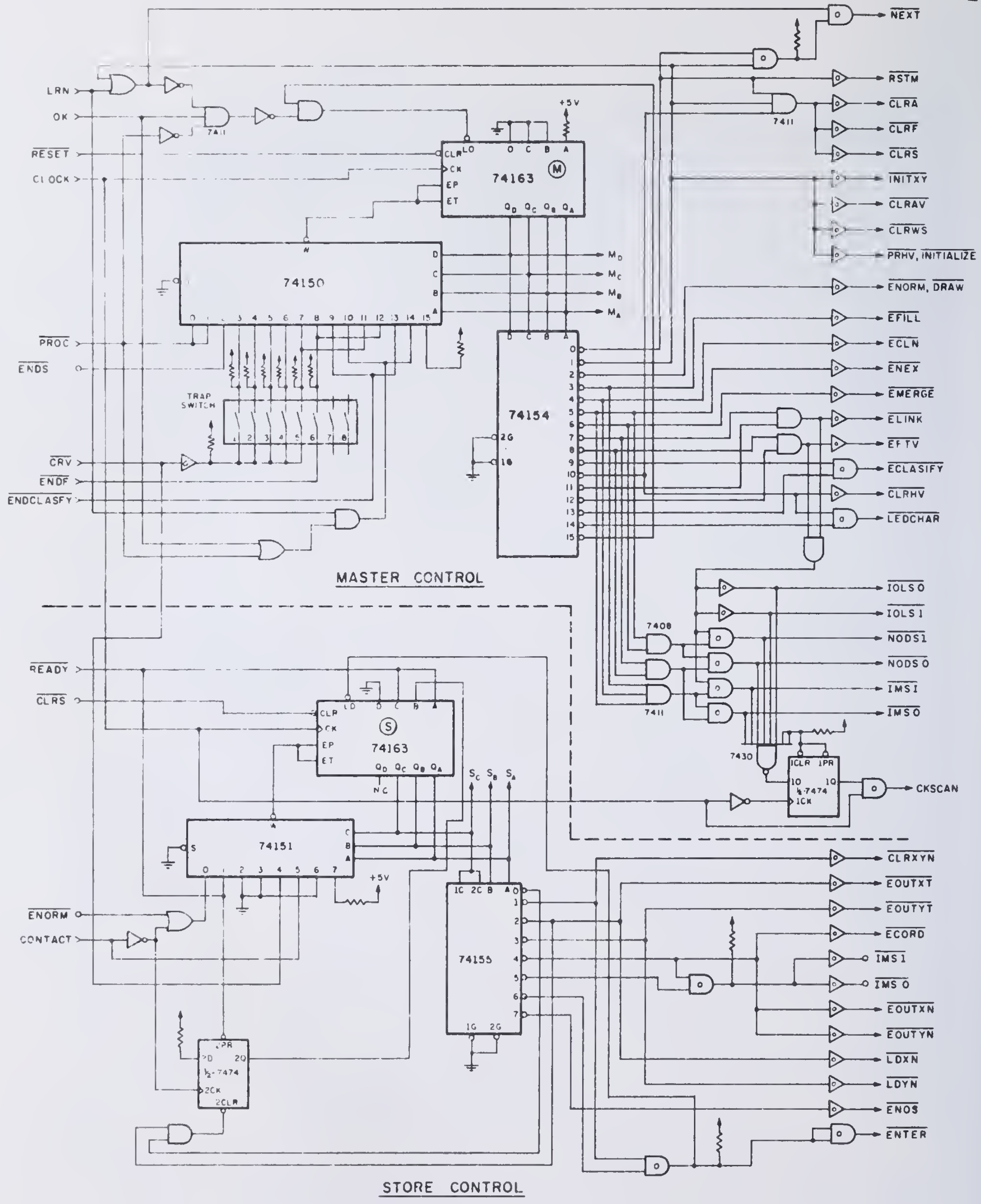


Figure B.7 MASTER and CONTROL Card

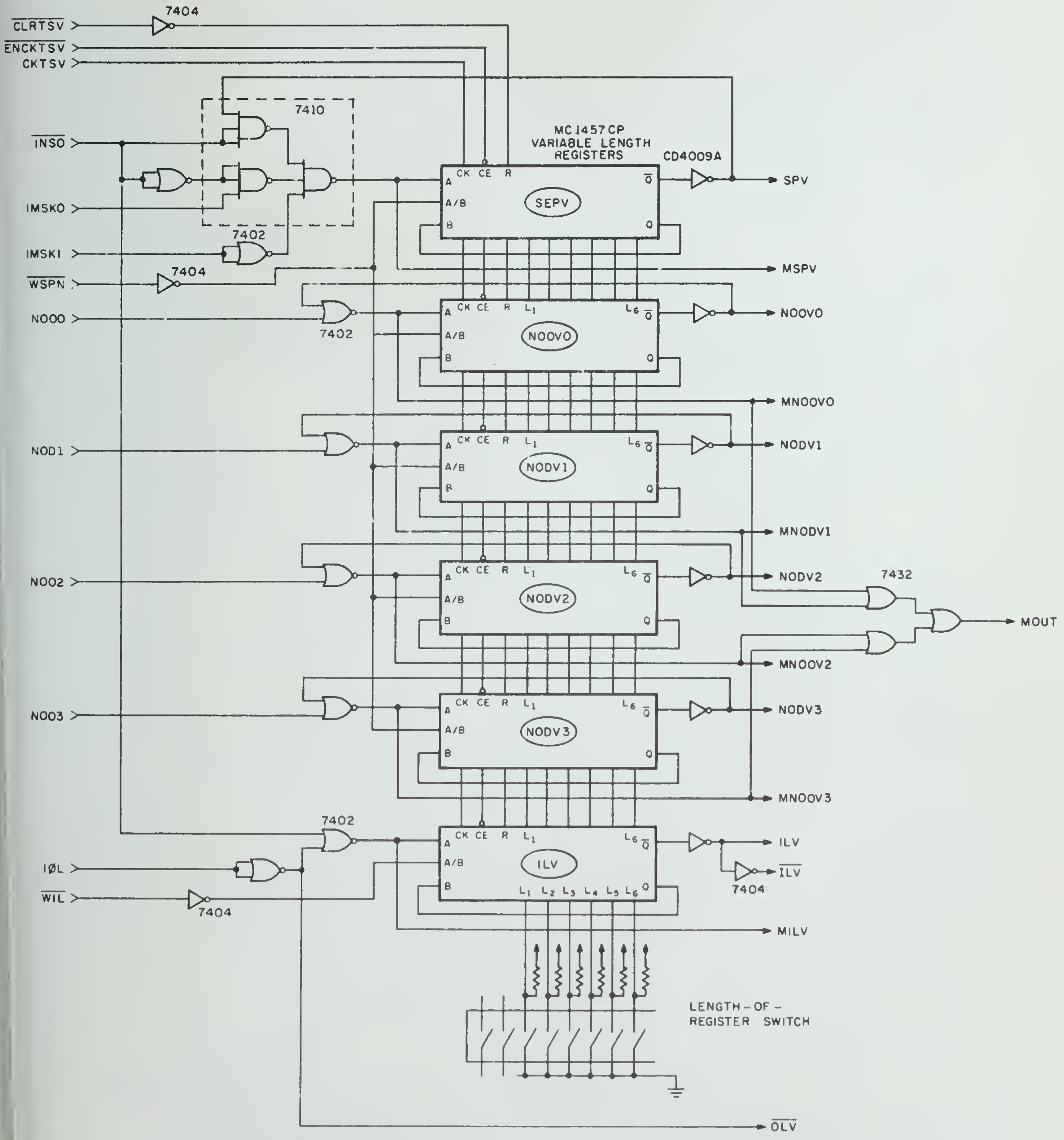


Figure B.8 TEMPORARY SEGMENT Module

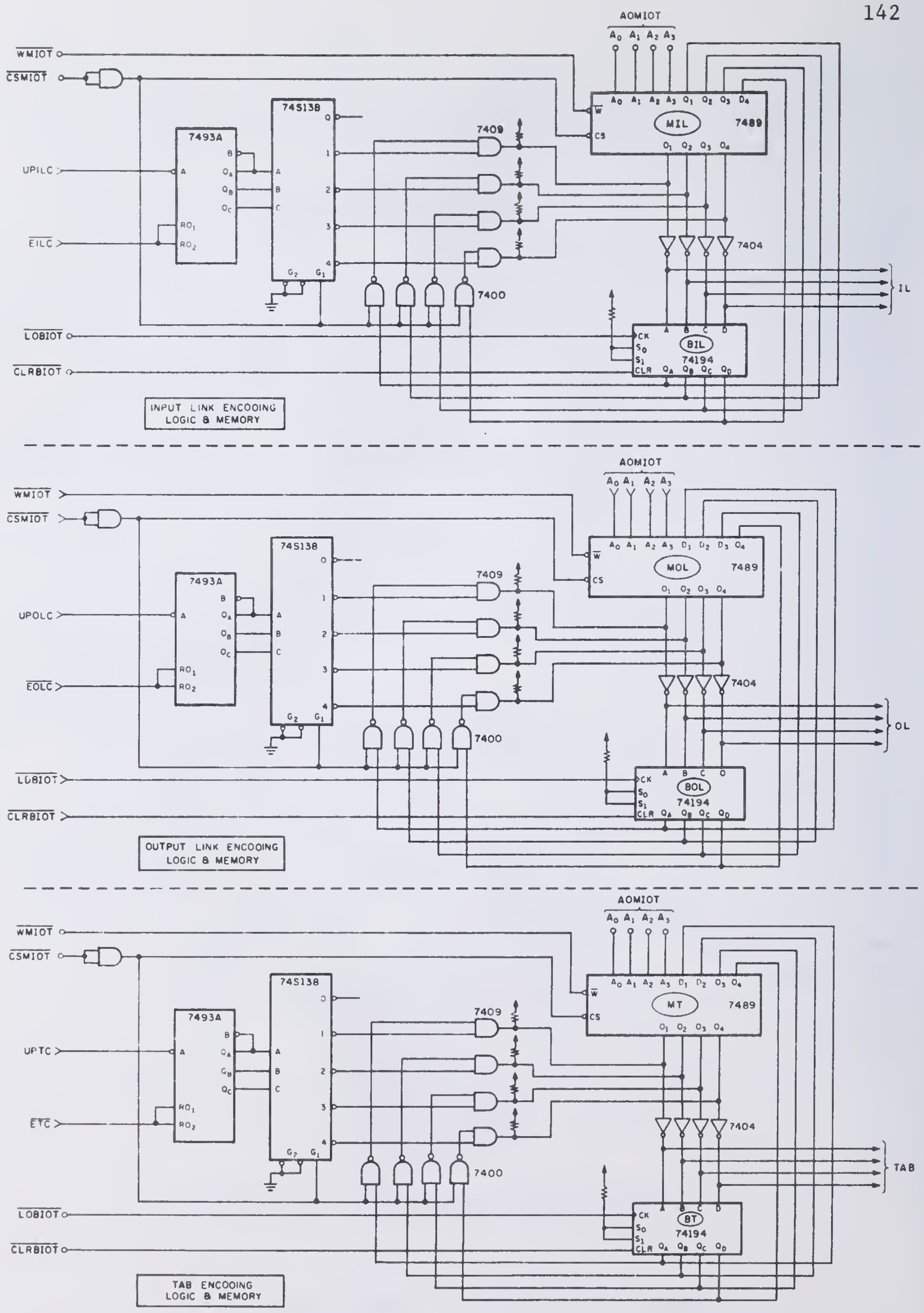


Figure B.9 FEATURE SUBVECTORS MEMORIES Module

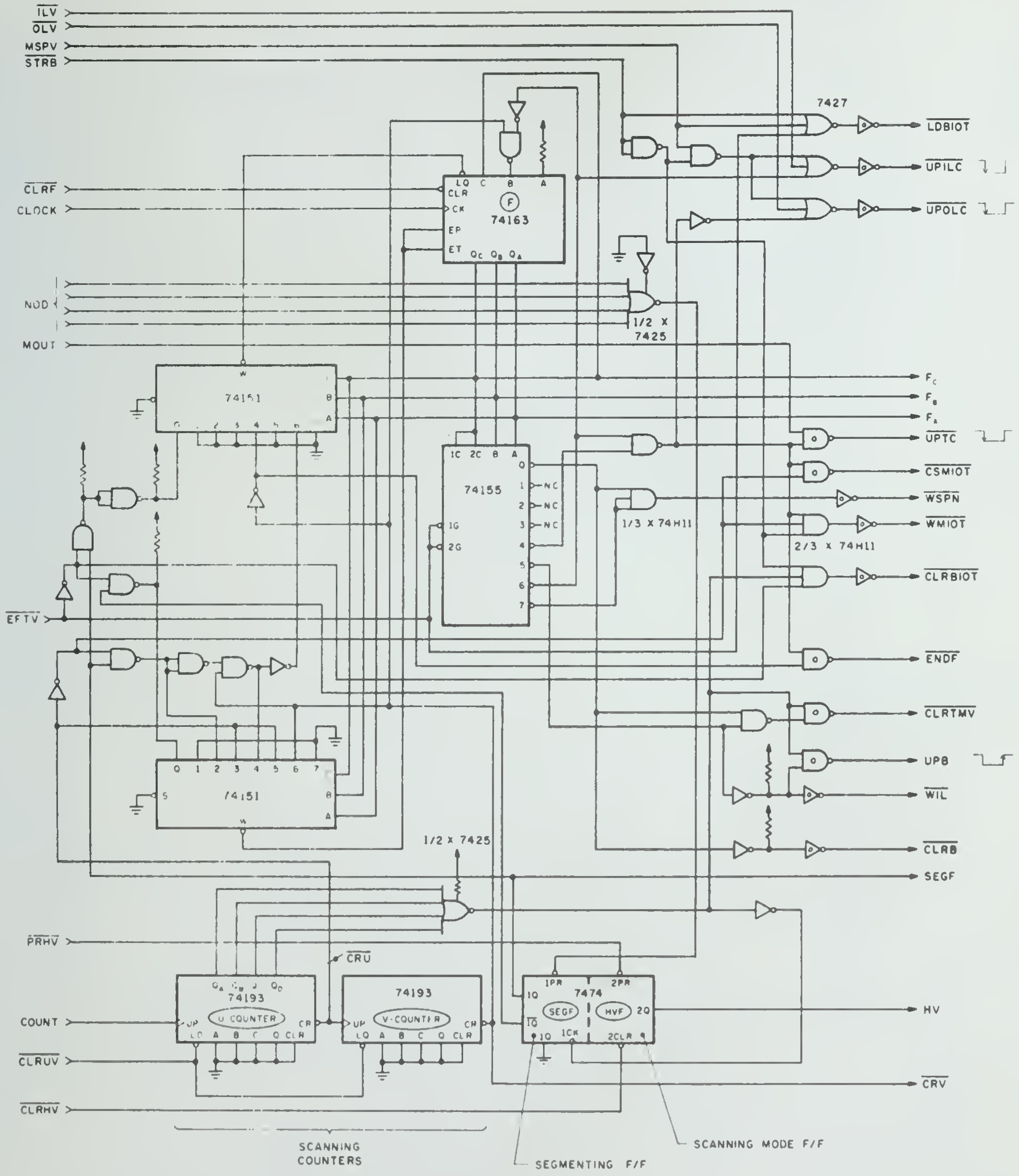


Figure B.10 F-CONTROL Card

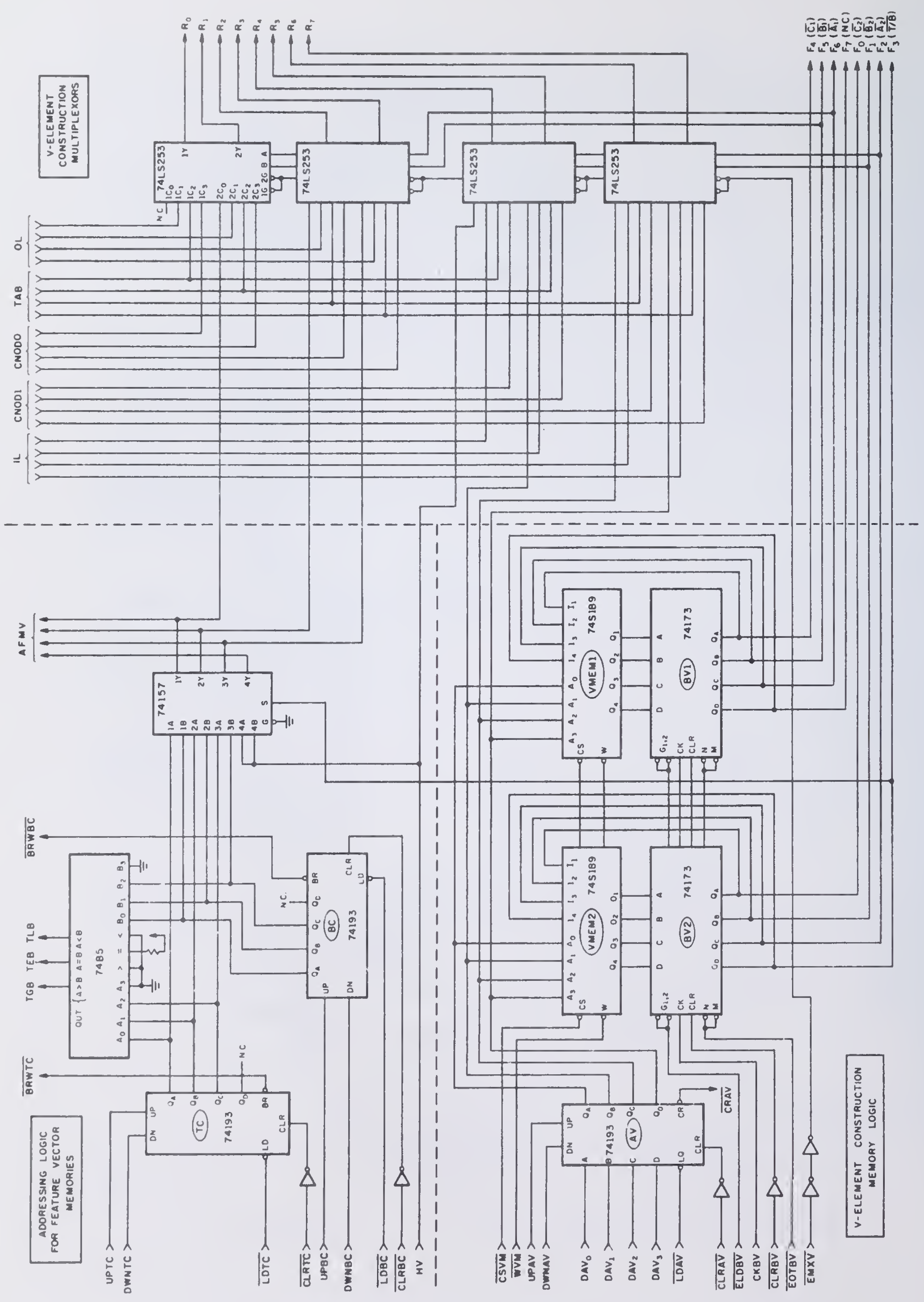


Figure B.11 V-ELEMENT CONSTRUCTION Module

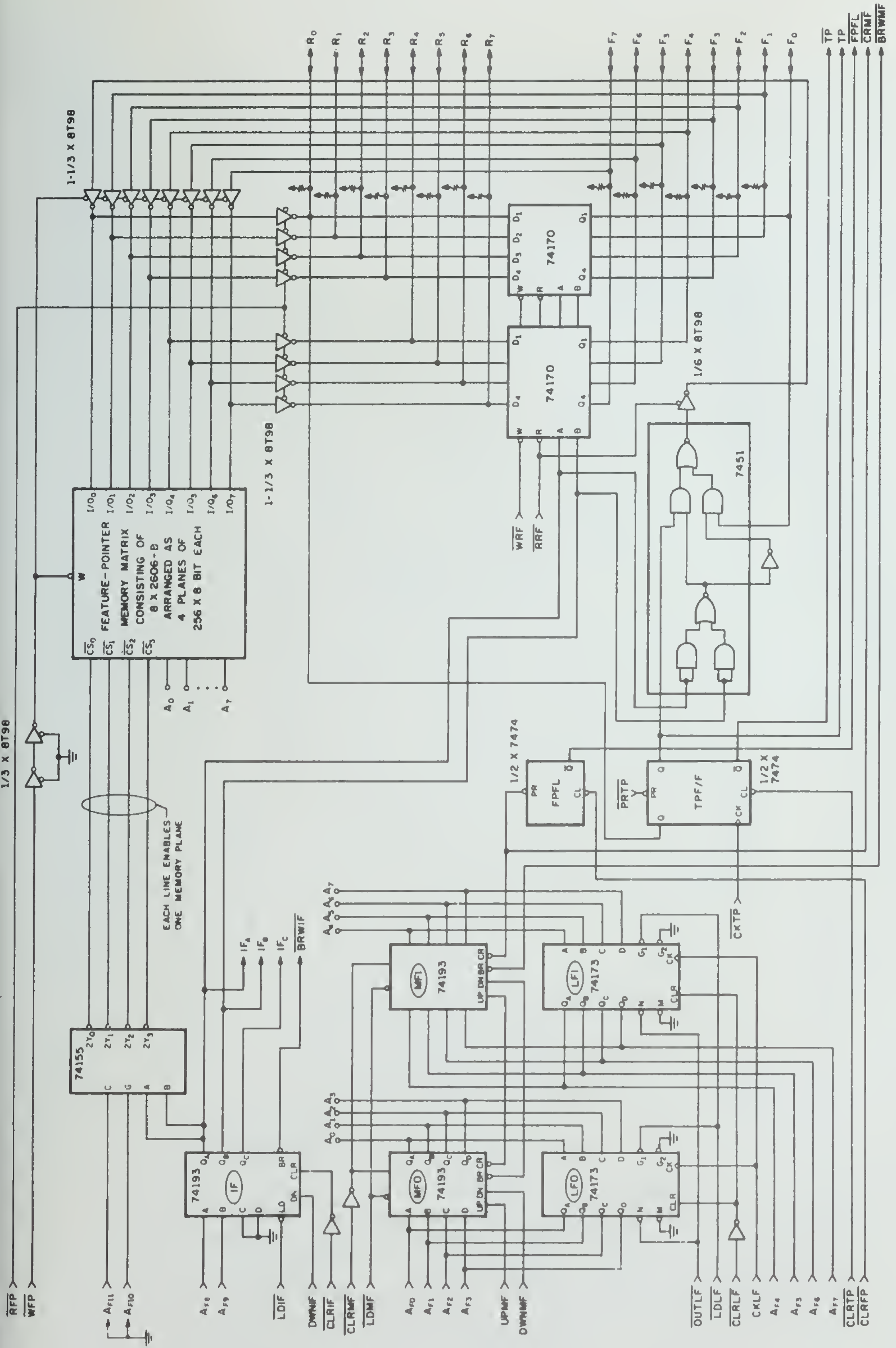


Figure B.12 FEATURE POINTER LIST Module

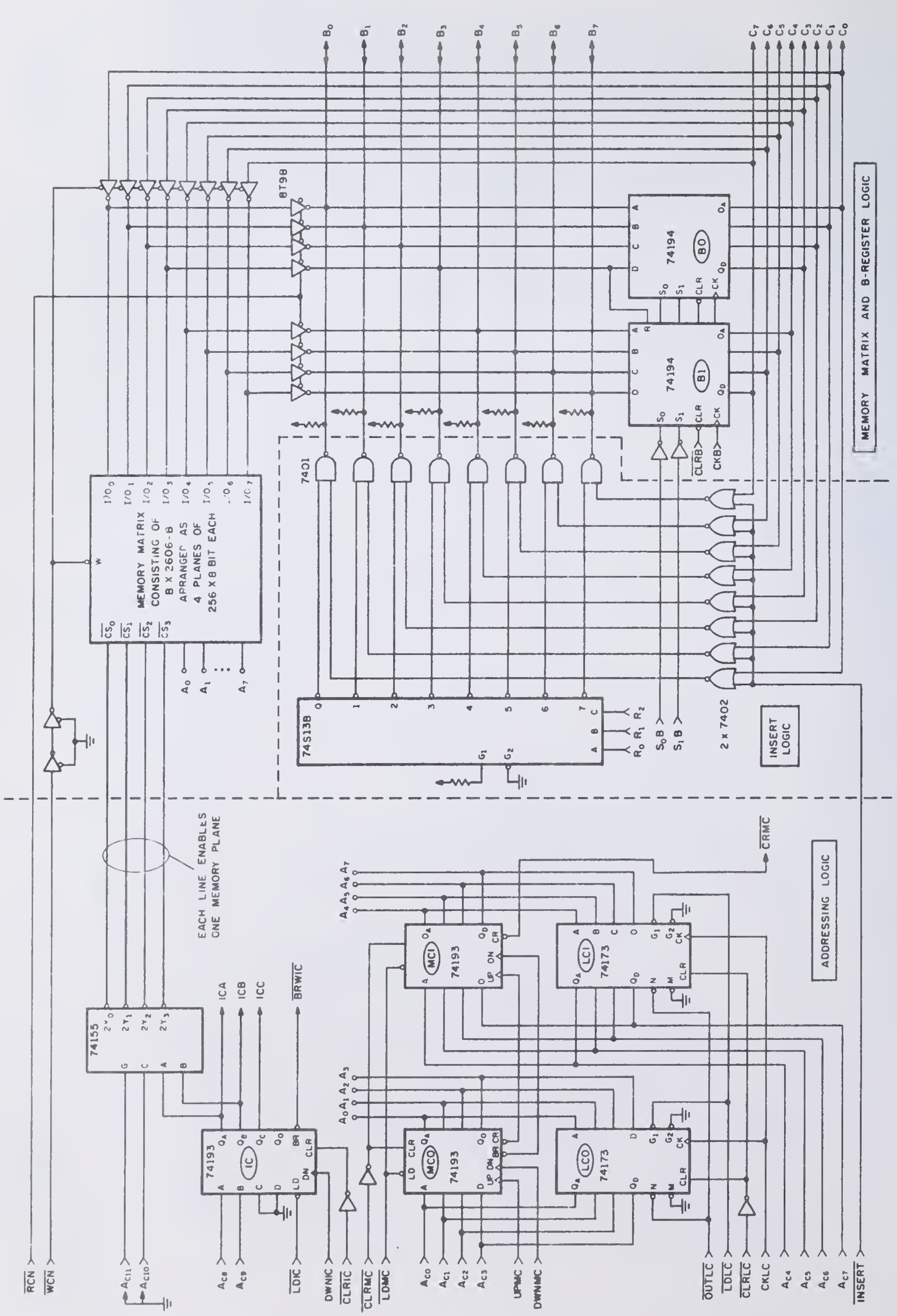


Figure B.13 CONFUSION MATRIX Module

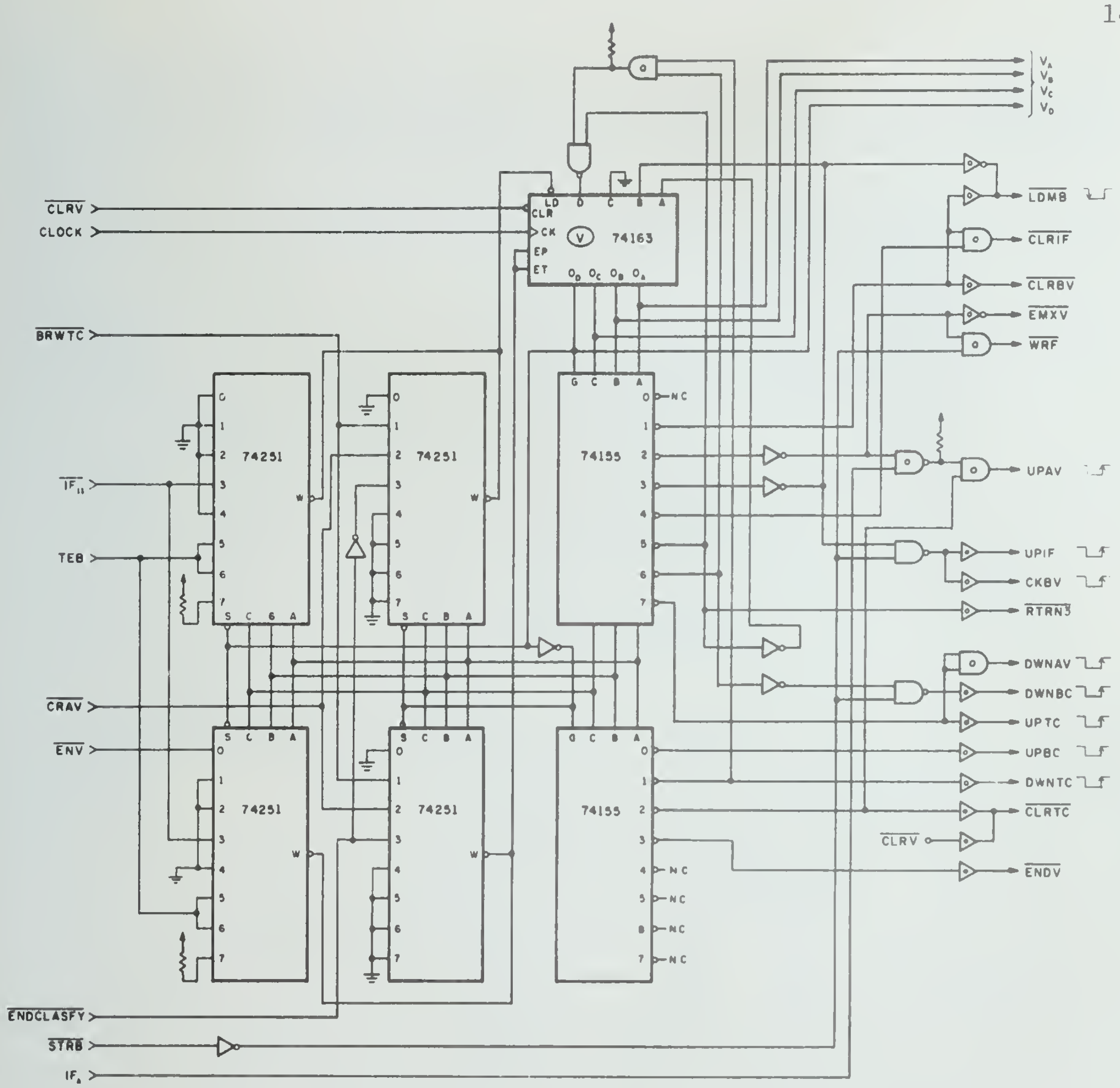


Figure B.14 V-CONTROL Card

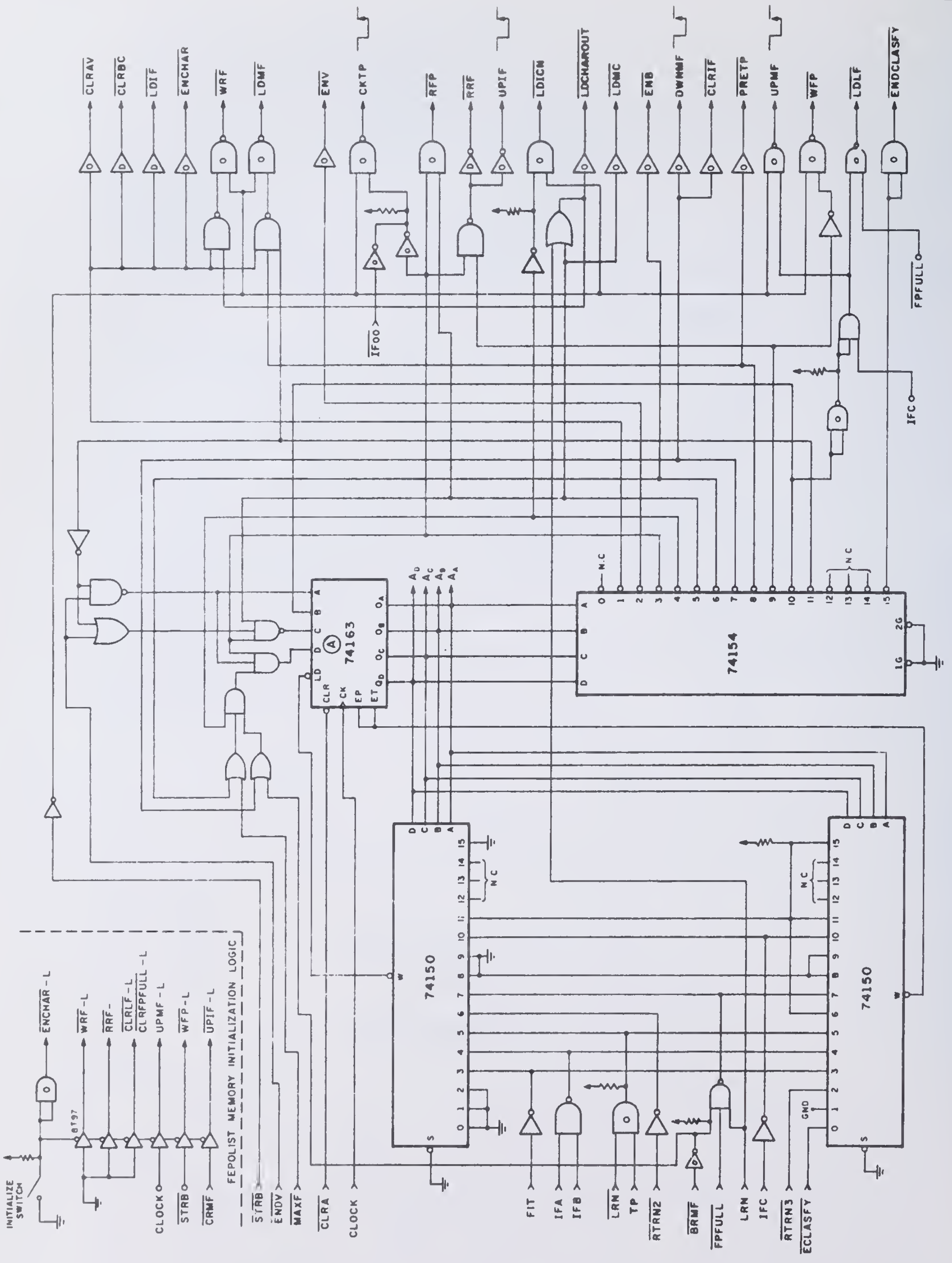


Figure B.15 A-CONTROL Card

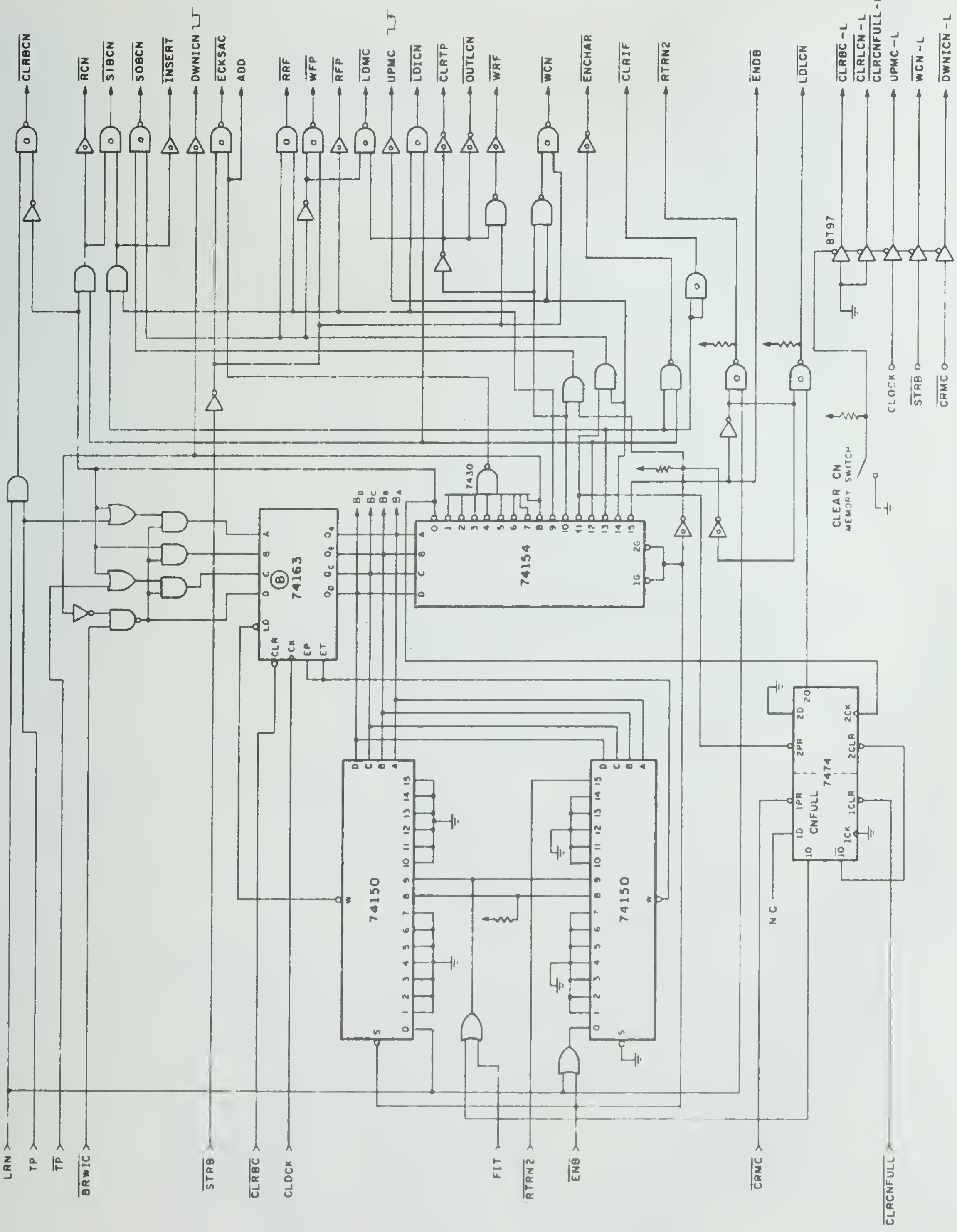


Figure B.16 B-CONTROL Card

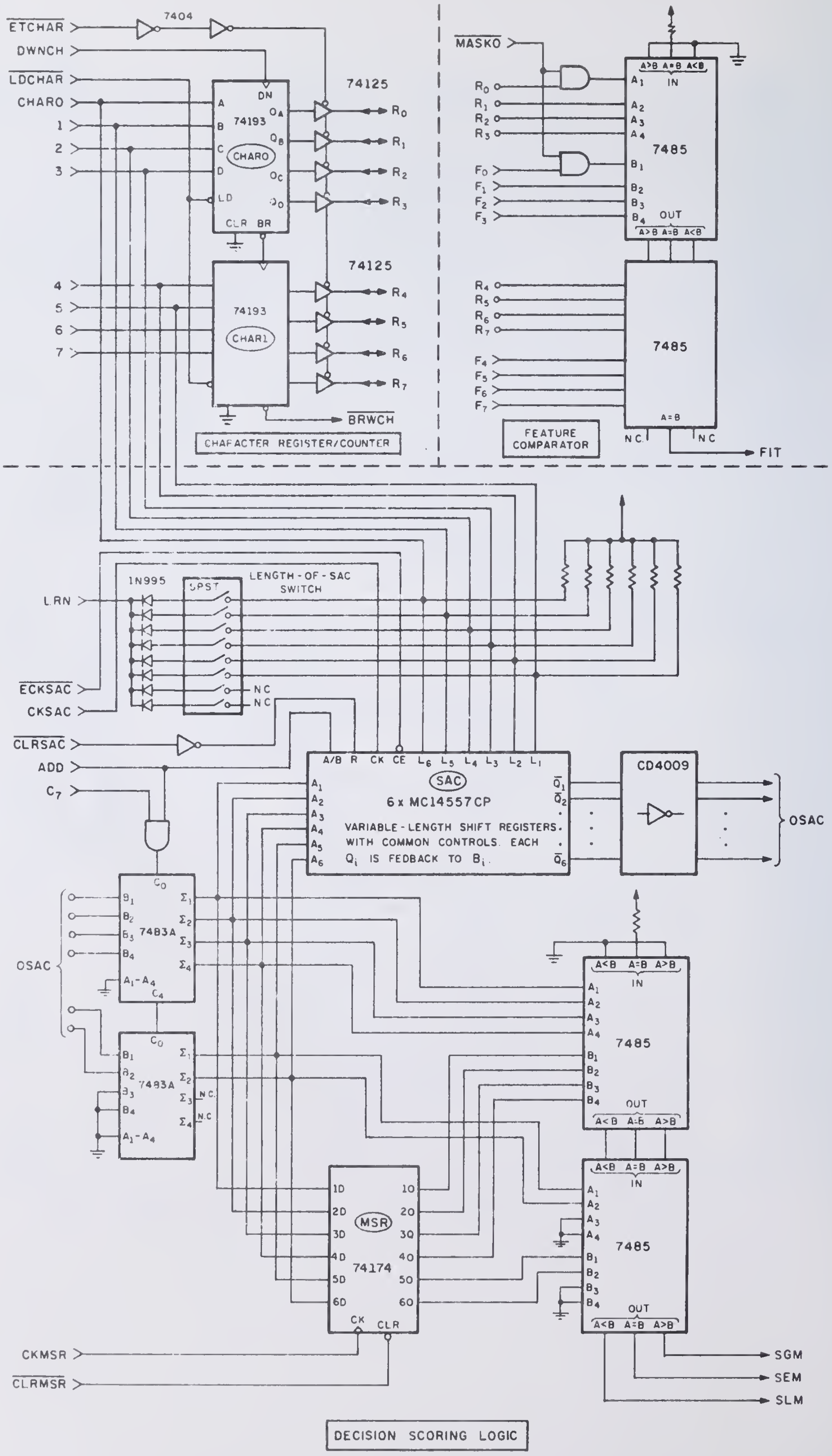


Figure B.17 DECISION Module

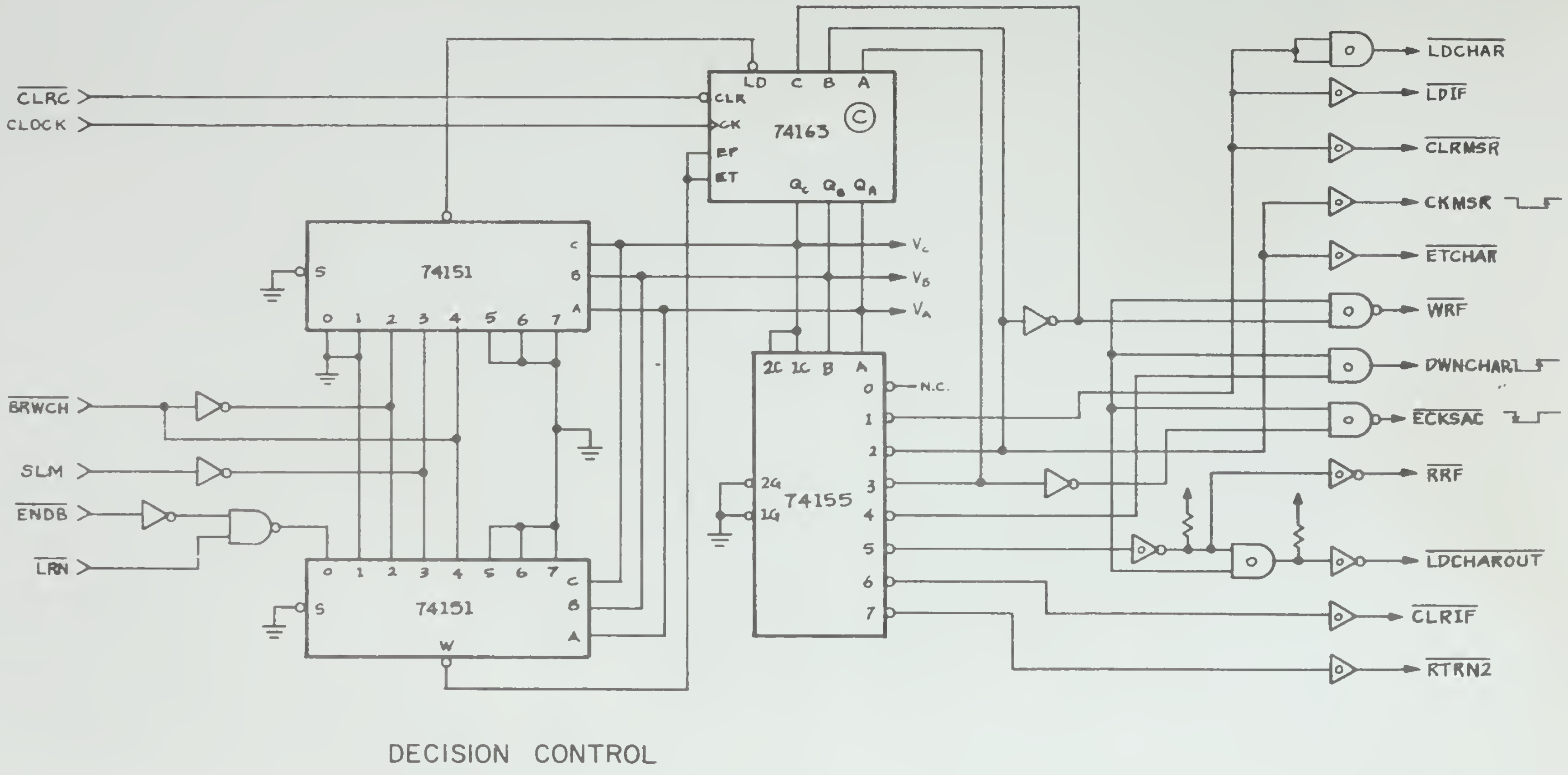


Figure B-18 C-CONTROL Card

VITA

Mohamed Taher El-Sonni was born in Alexandria, Egypt on May 12, 1944. He received his Bachelor of Science degree in Electrical Engineering (Communications Section) with grade Distinction, First Class of Honor from Alexandria University, Alexandria, Egypt in June 1966. He was a teaching assistant in the Electrical Engineering Department in the same university from October 1966 until February 1968. After working with the Broadcasting Services in Libya, he got a teaching assistantship from the Alfateh University in Tripoli, Libya in February 1969.

In September 1969, he joined the University of Illinois at Urbana-Champaign under a graduate study scholarship from the Libyan government. During the period from January 1972 until May 1975 he had served as research and teaching assistant.

He is an associate member of Sigma Xi and a student member of the Institute of Electrical and Electronics Engineers.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-78-944	2.	3. Recipient's Accession No.
4. Title and Subtitle TRAINABLE CHARACTER RECOGNITION INTERFACE COMPUTER (INCOM)			5. Report Date Oct. 1978
			6.
7. Author(s) Mohamed Taher Abdalla El-Sonni			8. Performing Organization Rept. No UIUCDCS-R-78-944
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No.
12. Sponsoring Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801			13. Type of Report & Period Covered Ph.D. Thesis
			14.
15. Supplementary Notes			

16. Abstracts

A trainable, real-time character recognition device has been designed and built. The visual feature extraction method is chosen as the most favorable computational approach around which the feature extractor is designed. A new method of local feature extraction is presented which uses a minimal size window and simple raster scanning of the character image. Merging rules are devised to reduce the number of these features by merging two features at a time. A method of dynamic segmentation of the character representations is introduced, in which the character is described as a collection of horizontal or vertical segments connected by links. A multiple description technique of the character segments make it possible to reduce the number of training characters and to use a simple data structure for the classification dictionary as well as a simple decision criterion for recognition. Experiments show the power of the described techniques.

17. Key Words and Document Analysis. 17a. Descriptors

Character Recognition
Trainable Machine
Feature Extraction
Node Extraction
Dynamic Segmentation
Scanning-Windowing Schemes

17b. Identifiers/Open-Ended Terms

7c. COSATI Field/Group

8. Availability Statement Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 152
	20. Security Class (This Page UNCLASSIFIED	22. Price